Current models severely underestimate future irrigated areas

Arnald Puy, Samuele Lo Piano and Andrea Saltelli

Contents

1 Preliminary steps

```
# PRELIMINARY STEPS ----------------------------------------------------

# Before starting the analysis, we define a function to load all R
# packages required in one go, and load them. We then create a function
# to read in all the spreadsheets of the excel file with the data
# needed for the analysis, and read in the data. Finally, we cast a
# function to define the theme of the plots that will be created in
# this work.

# Define function to read in all required libraries in one go:
loadPackages <- function(x) {
  for(i in x) {
    if(!require(i, character.only = TRUE)) {
      install.packages(i, dependencies = TRUE)
      library(i, character.only = TRUE)
    }
  }
}

# Load all required libraries:
loadPackages(c("data.table", "fitdistrplus", "fGarch", "readxl", "countrycode",
               "scales", "tidyverse", "cowplot", "mvoutlier", "complmrob",
               "randtoolbox","robustbase", "parallel", "smatr", "boot",
               "doParallel", "sensitivity", "wesanderson",
               "grid", "gridExtra", "NbClust"))

# install and load sensobol 0.2.1
PackageURL <- "https://cran.r-project.org/src/contrib/Archive/sensobol/sensobol_0.2.1.tar.gz"
install.packages(PackageURL, repos=NULL, type="source")
library(sensobol)

# Set checkpoint

dir.create(".checkpoint")
library("checkpoint")

checkpoint("2020-03-11",
           R.version ="3.6.1",
           checkpointLocation = getwd())

# Define function to read in all excel spreadsheets in one go:
readAll <- function(name, tibble = FALSE) {
  sheets <- excel_sheets(name)
  df <- lapply(sheets, function(y) read_excel(name,
                                              sheet = y))
```

```
  if(!tibble) df <- lapply(df, as.data.frame)
  names(df) <- sheets
  df
}

# Read in all excel spreadsheets:
df <- readAll("full.dataset2.xlsx") %>%
  lapply(., function(x) mutate_if(x, is.character, as.factor))

# Redefine column names for population.estimate spreadsheet
colnames(df$population.estimate) <- c("Estimate", "Continent", "Codes",
                                      paste0("Year.", 2015:2100))

# Create function for custom plot themes
theme_AP <- function() {
  theme_bw() +
    theme(aspect.ratio = 1,
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.background = element_rect(fill = "transparent",
                                           color = NA),
          legend.key = element_rect(fill = "transparent",
                                    color = NA))
}
```

2 Scaling relationships

2.1 Irrigated area versus population size

```
# PREPARE DATASETS ----------------------------------------------------

# Prepare population dataset
pop <- df$pop %>%
  # Select population between 1999-2012 to
  # get the mean and concur with (mean) values
  # for irrigated areas attested by Meier et al.
  # between 1999-2012
  select(Continent, Country, Year.1999:Year.2012) %>%
  gather(Year, Population, Year.1999:Year.2012) %>%
  # Multiply to get actual population
  mutate_at(vars(Population), funs(.*1000)) %>%
  separate(., Year, into = c("dummy", "Year")) %>%
  mutate_at(vars(Year), funs(as.numeric)) %>%
  select(-dummy)

# Obtain number codes for each country to ease merging
# with irrigated areas at the country level
```

```r
pop$Codes <- countrycode(pop$Country, origin = "country.name",
                         destination = "un")


# Create temporal list of data frame splitted by Dataset
temp <- df$meier %>%
  gather(Dataset, Area.irrigated, Meier.et.al.2018:Thenkabail.et.al.2009) %>%
  split(., .$Dataset)

# MERGE EACH DATASET WITH CORRESPONDING POPULATION VALUES ----------------

df.meier <- list()

for(i in names(temp)) {
  if(i == "Thenkabail.et.al.2009") {
    # Merge with population from 1999
    df.meier[[i]] <- pop %>%
      filter(Year == 1999) %>%
      inner_join(., temp[[i]],
                 by = c("Continent", "Country", "Codes"))
  }
  if(i == "Salmon.et.al.2015") {
    # Merge with population from 2005
    df.meier[[i]] <- pop %>%
      filter(Year == 2005) %>%
      inner_join(., temp[[i]],
                 by = c("Continent", "Country", "Codes"))
  }
  if(i == "Siebert.et.al.2013") {
    df.meier[[i]] <- pop %>%
      # Merge with mean population values 2000-2008
      spread(., Year, Population) %>%
      select(Continent, Country, `2000`:`2008`) %>%
      gather(Year, Population, `2000`:`2008`) %>%
      group_by(Country, Continent) %>%
      summarise(Population = mean(Population)) %>%
      mutate(Year = "2000.2008") %>%
      inner_join(., temp[[i]],
                 by = c("Continent", "Country"))
  } else {
    # For Meier et al.2018, Aquastat and FAOSTAT
    df.meier[[i]] <- pop %>%
      # Merge with mean population values 1999-2012
      group_by(Country, Continent) %>%
      summarise(Population = mean(Population)) %>%
      # Add dummy year column
      mutate(Year = "1999.2012") %>%
```

```
      inner_join(., temp[[i]],
                 by = c("Continent", "Country"))
  }
}

# Create dataset

cols <- c("Population", "Area.irrigated")

df.meier <- df.meier %>%
  rbindlist() %>%
  .[, .(Country, Continent, Codes,
    Area.irrigated, Population, Dataset)] %>%
  # Drop countries with no irrigated area
  .[!Area.irrigated == 0] %>%
  # Drop Oceania due to small sample size
  .[!Continent == "Oceania"] %>%
  .[, (cols):= .SD / 10^6, .SDcols = (cols)]
```

2.2 Irrigated area versus population density

```
# PLOT IRRIGATED AREAS AGAINST POPULATION DENSITY MEASURES ----------------------

# Read in dataset
density.population <- fread("density_population.csv")

# Rename variables
density.population <- density.population[, Variable:= ifelse(Variable %in%
                                                               "Total area of the country (excl

# Reduce dataset
density.population <- density.population[, c(2, 4, 5):= NULL]

# Spread
dens <- spread(density.population, Variable, Value)

cols <- c("Population", "Area.irrigated")

# Plot
inner_join(df.meier, dens, by = "Country") %>%
  data.table() %>%
  .[, `Area cultivable`:= Population / ((Area.cultivated * 1000) / 10^6)] %>%
  .[, `Area country`:= Population / ((Area.country * 1000) / 10^6)] %>%
  gather(parameter, value, `Area cultivable`:`Area country`) %>%
  ggplot(., aes(value, Area.irrigated,
                color = Continent)) +
  geom_point() +
```

```
  facet_grid(Dataset~parameter) +
  labs(x = "Population density (M/Mha)",
       y = "Irrigated area (Mha)") +
  scale_x_log10(labels = trans_format("log10", math_format(10^.x))) +
  scale_y_log10(labels = trans_format("log10", math_format(10^.x))) +
  theme_bw() +
  theme(legend.position = "top",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.background = element_rect(fill = "transparent",
                                         color = NA),
        legend.key = element_rect(fill = "transparent",
                                  color = NA))
```

2.3 Plot irrigated area versus population size / population density

```
# PLOT SCATTER PLOT OF IRRIGATED AREA AND POPULATION ---------------------

df.meier %>%
  ggplot(., aes(Population, Area.irrigated,
                color = Continent)) +
  geom_point() +
  facet_wrap(~Dataset) +
  labs(x = "Population (M)",
       y = "Irrigated area (Mha)") +
  scale_x_log10(labels = trans_format("log10", math_format(10^.x))) +
  scale_y_log10(labels = trans_format("log10", math_format(10^.x))) +
  theme_AP() +
  theme(legend.position = "top")
```

2.4 Irrigated area versus water withdrawal / water requirement

```
# READ IN AQUASTAT DATA SET ----------------------------------------------------

aquastat <- fread("aquastat.csv",
                  nrows = 3562)

# ARRANGE AQUASTAT DATA SET ----------------------------------------------------

aquastat <- setnames(aquastat, c("Area", "Variable Name"),
                     c("Country", "Variable"))

cols <- c("Country", "Variable")

aquastat <- aquastat[, (cols):= lapply(.SD, factor), .SDcols = cols] %>%
  .[, Variable:= fct_recode(Variable, "Water.withdrawal" = "Irrigation water withdrawal",
                            "Water.requirement" = "Irrigation water requirement")] %>%
```

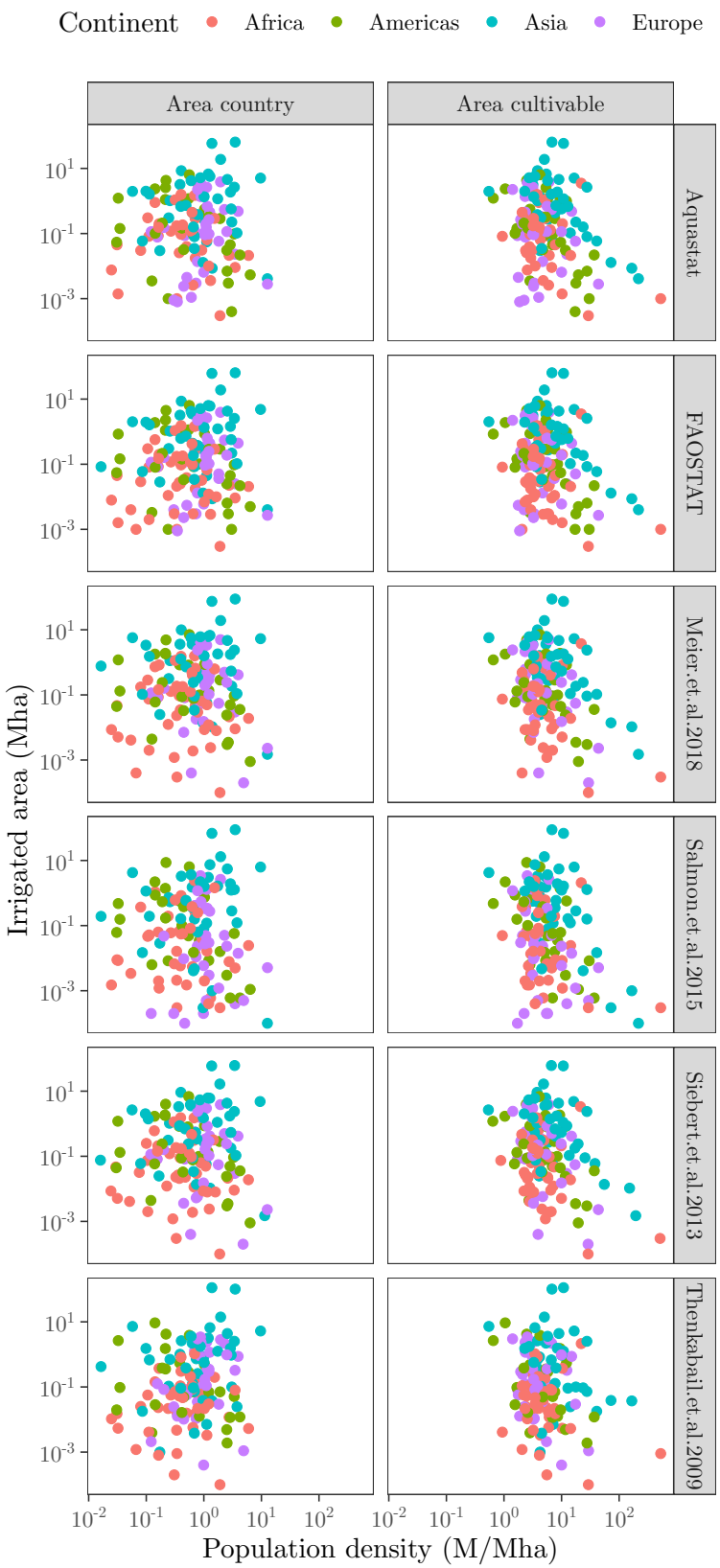


Figure 1: Scatter plots of measures of population density against irrigated areas.

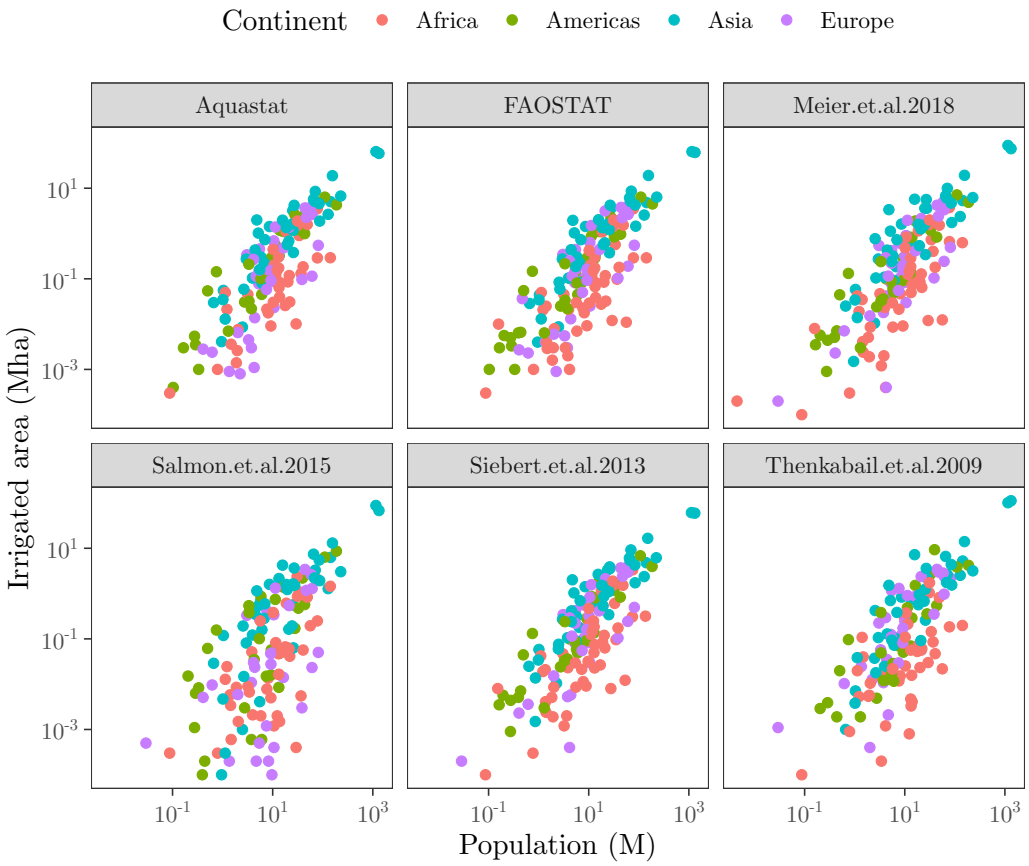


Figure 2: Scatter plots of irrigated areas against population. The strip label indicates the data set used to plot the values for irrigated areas [@FAO2016b; @FAO2017a; @Meier2018; @Salmon2015; @Siebert2013; @Thenkabail2009]. All those data sets have been compiled and studied by @Meier2018. The population data was retrieved from the @UnitedNations2017c.

```
  .[Variable == "Water.withdrawal" |Variable == "Water.requirement",
    .(Country, Variable, Year, Value)] %>%
  .[!c(Year <= 1999 | Year >= 2012)] # Retain only years 1999-2012
```

```
# FUNCTIONS TO CODE ------------------------------------------------------

getCodes <- function(x) countrycode(x, origin = "country.name", destination = "un")
getContinent <- function(x) countrycode(x, origin = "country.name", destination = "continent")
getCountry <- function(x) countrycode(x, origin = "un", destination = "country.name")

addAll <- function(dt, dataset) {
  if(is.data.table(dt) == FALSE) {
    setDT(dt)
  }
  dt[, Codes:= lapply(.SD, getCodes), .SDcols = "Country"] %>%
    .[, Continent:= lapply(.SD, getContinent), .SDcols = "Country"] %>%
    .[, Country:= lapply(.SD, getCountry), .SDcols = "Codes"] %>%
    .[, Dataset:= dataset]
}
```

```
# CODE COUNTRY AND CONTINENT ---------------------------------------------

aquastat <- addAll(aquastat, "Aquastat")

aquastat.dt <- spread(aquastat, Variable, Value)[
  !c(Year <= 1999 | Year >= 2012) # Retain only years 1999-2012
  ]

# READ IN TABLE 4 DATA SET -----------------------------------------------

table4 <- fread("table_4.csv",
                skip = 3,
                nrows = 167) %>%
  .[, c(2, 5, 7):= NULL]

# CODE COUNTRY AND CONTINENT ---------------------------------------------

table4.dt <- addAll(table4, "Table.4") %>%
  .[!Continent == "Oceania"] %>%
  .[, .(Country, Year, Codes, Continent, Dataset, Water.requirement, Water.withdrawal)] %>%
  .[!c(Year <= 1999 | Year >= 2012)] # Retain only years 1999-2012

# MERGE AQUASTAT AND TABLE 4 DATASETS ------------------------------------

water.dt <- rbind(aquastat.dt, table4.dt) %>%
  melt(., measure.vars = c("Water.requirement", "Water.withdrawal")) %>%
  .[, .(Max = max(value, na.rm = TRUE),
        Min = min(value, na.rm = TRUE)),
```

```
    by = .(Continent, variable, Country)]
```

```
## Warning in gmax(value, na.rm = TRUE): No non-missing values found in at
## least one group. Returning '-Inf' for such groups to be consistent with
## base
```

```
## Warning in gmin(value, na.rm = TRUE): No non-missing values found in at
## least one group. Returning 'Inf' for such groups to be consistent with base
```

```
# Transform Inf values in NA
is.na(water.dt) <- do.call(cbind,lapply(water.dt,
                                        is.infinite))

# READ IN MEIER ET AL. DATASET ------------------------------------------------

meier.dt <- df$meier %>%
  data.table() %>%
  .[!Continent == "Oceania"] %>%
  .[, (4:9):= lapply(.SD, function(x) x / 10^6), .SDcols = (4:9)]

# CODE COUNTRY AND CONTINENT --------------------------------------------------

meier.dt <- addAll(meier.dt, "Meier")[, Dataset:= NULL]
```

```
## Warning in countrycode(x, origin = "country.name", destination = "un"): Some values were not
```

```
# CREATE FINAL DATASET --------------------------------------------------------

dt_water <- melt(water.dt, measure.vars = c("Max", "Min"),
                 variable.name = "Stat") %>%
  .[, .(Value = mean(value, na.rm = TRUE)),
    by = .(Continent, variable, Country)] %>%
  .[meier.dt, on = c("Country", "Continent")] %>%
  .[!variable %in% NA] %>% # Remove rows in variable with NA
  melt(., measure.vars = c(6:11),
       variable.name = "Dataset",
       value.name = "Area.irrigated") %>%
  .[, Dataset:= factor(Dataset, levels = c("Aquastat", "FAOSTAT", "Meier.et.al.2018",
                                           "Salmon.et.al.2015", "Siebert.et.al.2013",
                                           "Thenkabail.et.al.2009"))]

dt.full <- dt_water[variable == "Water.requirement"] %>%
  .[df.meier, on = c("Continent", "Country" ,"Dataset", "Codes", "Area.irrigated")] %>%
  setnames(., "Value", "Water") %>%
  .[, .(Continent, Country, Codes, Dataset, Area.irrigated, Population, Water)]
```

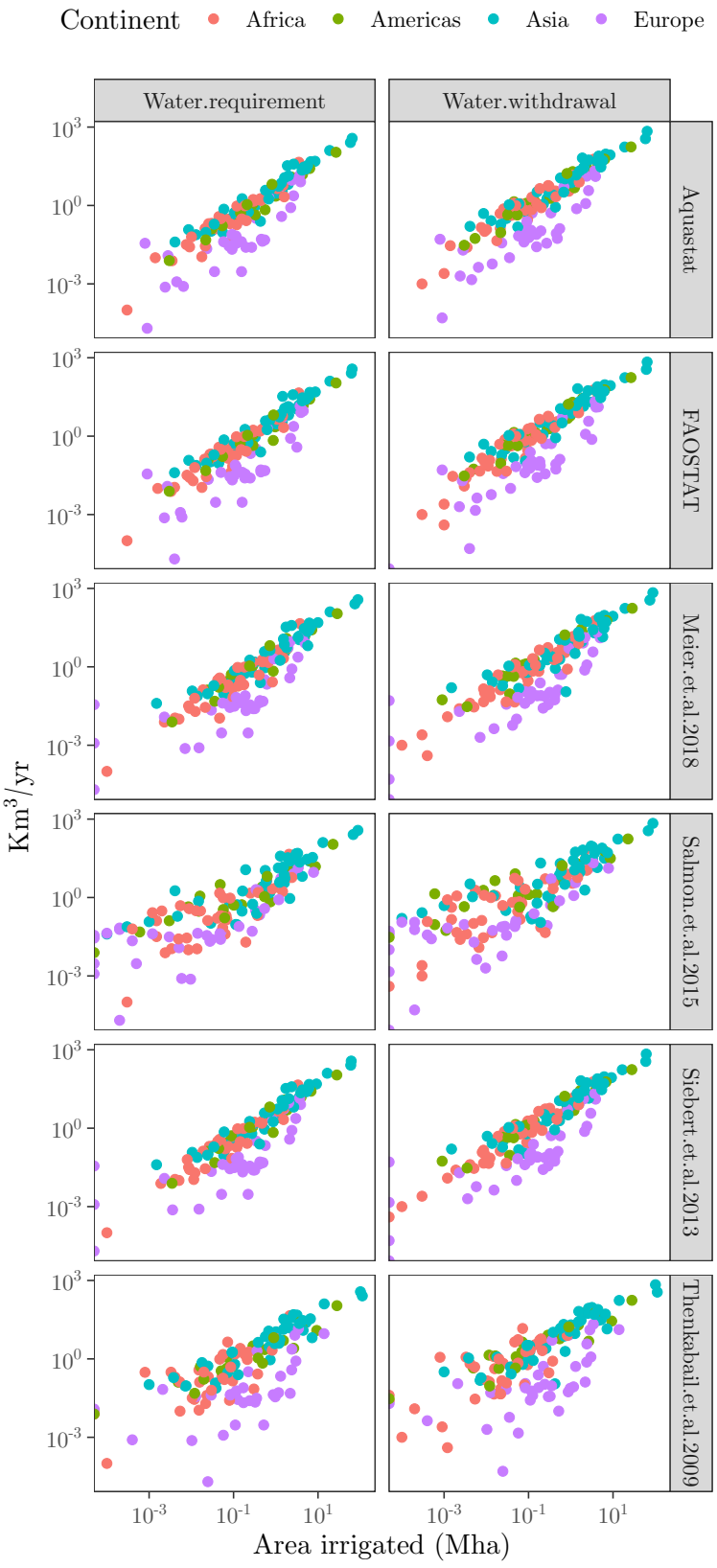
2.5 Plot irrigated area versus water withdrawal / water requirement

```
# PLOT WATER VARIABLES AGAINST IRRIGATED AREAS ---------------------------

ggplot(dt_water, aes(Area.irrigated, Value,
                     color = Continent)) +
  geom_point() +
  scale_x_log10(labels = trans_format("log10", math_format(10^.x))) +
  scale_y_log10(labels = trans_format("log10", math_format(10^.x))) +
  labs(x = "Area irrigated (Mha)",
       y = expression(Km^3/yr)) +
  facet_grid(Dataset ~ variable) +
  theme_bw() +
  theme(legend.position = "top",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.background = element_rect(fill = "transparent",
                                         color = NA),
        legend.key = element_rect(fill = "transparent",
                                  color = NA))
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 113 rows containing missing values (geom_point).
```


Continent
Africa
Americas
Asia
Europe
Water.requirement
Water.withdrawal
Aquastat
FAOSTAT
Meier.et.al.2018
Salmon.et.al.2015
Siebert.et.al.2013
Thenkabail.et.al.2009
Km3/yr
Area irrigated (Mha)

3 Uncertainty in irrigated areas

```
# PLOT DIFFERENCES IN THE MEASUREMENT OF IRRIGATED AREAS
# AS A FUNCTION OF DATASET (CONTINENTAL) (Figure 2) ---------------------

# Create data frame with total irrigated areas per dataset
total <- df$meier %>%
  filter(!Continent == "Oceania") %>%
  gather(Dataset, Value, Meier.et.al.2018:Thenkabail.et.al.2009) %>%
  group_by(Dataset) %>%
  summarise(Total = sum(Value, na.rm = T) / 10^6) %>%
  data.frame()

# Bar plot with continental and total irrigated areas per dataset
df$meier %>%
  gather(Dataset, Value, Meier.et.al.2018:Thenkabail.et.al.2009) %>%
  filter(!Continent == "Oceania") %>%
  group_by(Continent, Dataset) %>%
  summarise(Total = sum(Value, na.rm = T) / 10^6) %>%
  ggplot(., aes(Continent, Total)) +
  geom_bar(stat = "identity") +
  geom_text(data = total, aes(label = paste("Total: ",
                                            round(Total, digits = 2),
                                            " Mha",
                                            sep = ""),
                              group = Dataset),
            x = 4,
            y = 160,
            inherit.aes = FALSE,
            size = 3) +
  scale_y_continuous(breaks = pretty_breaks(n = 2)) +
  coord_flip() +
  labs(x = "Continent",
       y = "Irrigated area (Mha)") +
  facet_wrap(~Dataset) +
  theme_bw() +
  theme(aspect.ratio = 1,
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```

```
# DIFFERENCES IN THE MEASUREMENT OF IRRIGATED AREAS
# AS A FUNCTION OF DATASET (COUNTRY) ------------------------------------------

temp <- df$meier %>%
  gather(Dataset, Value, 4:ncol(.)) %>%
  filter(!c(Value == 0 |
              Continent == "Oceania")) %>%
  mutate(Country = fct_recode(Country,
```

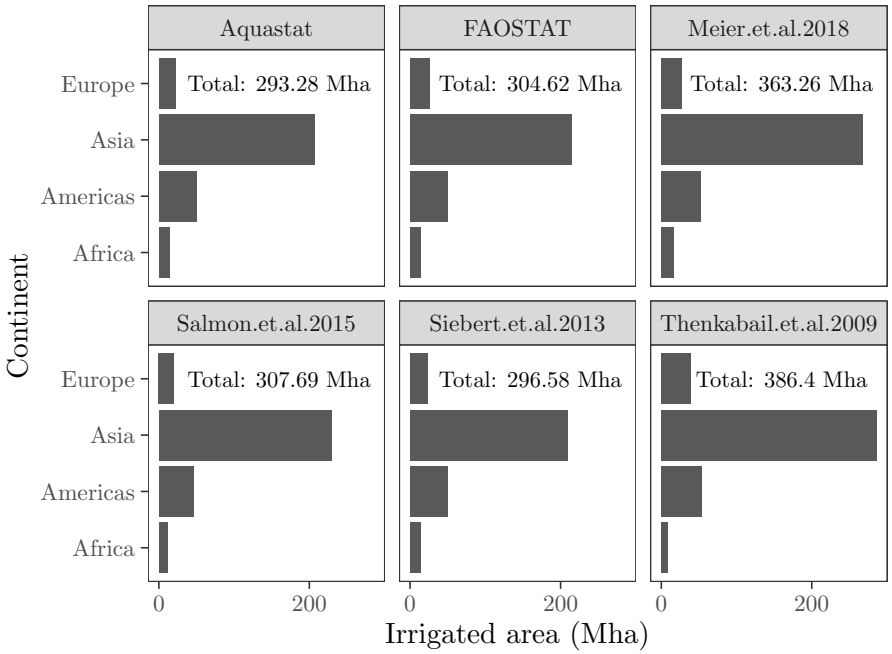


Figure 3: The extension of irrigation documented by different authors and institutions. FAO (2016) and FAOSTAT (FAO 2017) provide official values based on national surveys, census and statistics. Siebert et al. (2013) merges FAOSTAT and Aquastat values with independent maps and remote sensing imagery. Salmon et al. (2015) integrates national and subnational surveys with remote sensing and gridded climate data sets. Thenkabail et al. (2009) relies on remote sensing, Google Earth, and ground control points. Meier, Zabel, and Mauser (2018) downscales the map by Siebert et al. (2013) and uses multi-temporal normalized difference vegetation indexes with agricultural suitability data. The data was retrieved from Meier, Zabel, and Mauser (2018).

```
                                "Congo" = "Democratic Republic of the Congo",
                                "Tanzania" = "United Republic of Tanzania",
                                "Iran" = "Iran (Islamic Republic of)",
                                "Korea (DPR)" = "Korea, Democratic People's Republic of",
                                "Lao" = "Lao People's Democratic Republic",
                                "Macedonia" = "The former Yugoslav Republic of Macedonia")) %>%
  mutate_at(vars(Value), funs(. / 10^6)) %>%
  droplevels() %>%
  split(., .$Continent)

gg <- list()
for(i in seq(temp)) {
  gg[[i]] <- ggplot(temp[[i]], aes(reorder(Country, Value), Value)) +
    geom_point(stat = "identity", aes(color = Dataset)) +
    scale_y_log10(  breaks = trans_breaks("log10", function(x) 10^x),
                    labels = trans_format("log10", math_format(10^.x))) +
    coord_flip() +
    scale_color_manual(name = "Dataset",
                       labels = c("Aquastat", "FAOSTAT", "Thenkabail et al. 2009",
                                  "Siebert et al. 2013", "Salmon et al. 2015",
                                  "Meier et al. 2018"),
                       values = c("yellowgreen", "seagreen4", "magenta3",
                                  "sienna3", "turquoise2", "khaki3")) +
    labs(y = "Irrigated area (Mha)",
         x = "") +
    facet_wrap(~Continent,
               scales = "free_y") +
    theme_bw() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.position = "none",
          plot.margin = margin(t = -4.5,
                               unit = "cm"))
}

# Extract legend
legend <- get_legend(gg[[1]] + theme(legend.position = "top"))
```

```
# PLOT FOR AFRICA AND THE AMERICAS -----------------------------------

first <- plot_grid(gg[[1]], gg[[2]], ncol = 2)
plot_grid(legend, first, ncol = 1)
```

```
# PLOT FOR ASIA AND EUROPE -------------------------------------------

second <- plot_grid(gg[[3]], gg[[4]], ncol = 2)
plot_grid(legend, second, ncol = 1)
```

Figure 4: Extension of irrigation at the country level. The data was retrieved from Meier, Zabel, and Mauser (2018).

Figure 5: Extension of irrigation at the country level. The data was retrieved from Meier, Zabel, and Mauser (2018).

4 Presence of outliers

```
# CHECK WHETHER THERE ARE OUTLIERS IN AREA.IRRIGATED VS. POPULATION
# AND AREA.IRRIGATED VS WATER REQUIRED FOR IRRIGATION ------------------------

# Create datasets
temp <- dt_water[variable == "Water.requirement"] %>%
  .[df.meier, on = c("Continent", "Country" ,"Dataset", "Codes", "Area.irrigated")] %>%
  mutate_at(vars(Area.irrigated, Population, Value), funs(log10)) %>%
  data.table()

# Calculate Mahalabobis distances (robust and classic) for each
# continent and dataset

cols <- c("Population", "Area.irrigated")

temp1 <- temp[, dd.plot(.SD), .SDcols = cols, by = .(Continent, Dataset)] %>%
  .[, Class:= "Population"]

temp2 <- temp %>%
  na.omit() %>%
  .[, dd.plot(.SD), .SDcols = c("Value", "Area.irrigated"),
                  by = .(Continent, Dataset)] %>%
  .[, Class:= "Water"]

out <- rbind(temp1, temp2)

# Extract number of outliers per continent and dataset
out.n <- out[, .(Outliers = sum(outliers == TRUE)),
             by = .(Continent, Dataset, Class)] %>%
  .[order(Continent)]

# Extract maximum values for Mahalanobis distances
# (robust and classic)
out.md <- out[, .(max.Mahalanobis.classic = max(md.cla),
                  max.Mahalanobis.robust = max(md.rob)),
              by = .(Dataset, Continent, Class)]

# Merge both datasets
out.df <- out.n[out.md, on = c("Dataset", "Continent", "Class")] %>%
  .[order(Continent)]
```

```
# EXPORT OUTLIERS DATASET ----------------------------------------------

fwrite(out.df, "out.df.csv")
```

```
# ARRANGE TO PLOT RESULTS ----------------------------------------------------
```

```
temp <- out %>%
  split(., .$Class)

gg <- list()
for(i in names(temp)) {
  gg[[i]] <-   ggplot(temp[[i]], aes(md.cla, md.rob,
                                     color = outliers)) +
    geom_point() +
    scale_colour_manual(name = "Outlier",
                        values = setNames(c("black", "red"),
                                          c(FALSE, TRUE))) +
    facet_grid(Dataset~Continent) +
    labs(x = "Mahalanobis distance",
         y = "Robust distance") +
    theme_AP() +
    theme(legend.position = "top")
}
```

```
# PLOT MAHALANOBIS DISTANCES FOR AREA IRRIGATED VERSUS POPULATION -------------

plot(gg[["Population"]])
```

```
# PLOT MAHALANOBIS DISTANCES FOR AREA IRRIGATED VERSUS
# IRRIGATION WATER REQUIREMENT ------------------------------------------------

plot(gg[["Water"]])
```

5 Estimation of the model parameters

5.1 Irrigated area baseline values (Y_0)}

```
# CALCULATE FOR EACH CONTINENT THE MAXIMUM AND MINIMUM
# EXTENSION OF IRRIGATED AREAS ------------------------------------------

total.area.irrigated <- df$meier %>%
  data.table() %>%
  melt(., measure.vars = c(4:9),
       variable.name = "Dataset",
       value.name = "Value") %>%
  .[, Value:= Value / 10 ^6] %>%
  .[, .(Total = sum(Value, na.rm = T)), .(Dataset, Continent)] %>%
  .[, .(min = min(Total), max = max(Total)), Continent] %>%
  .[!Continent == "Oceania"] %>%
  split(., .$Continent, drop = TRUE)
```

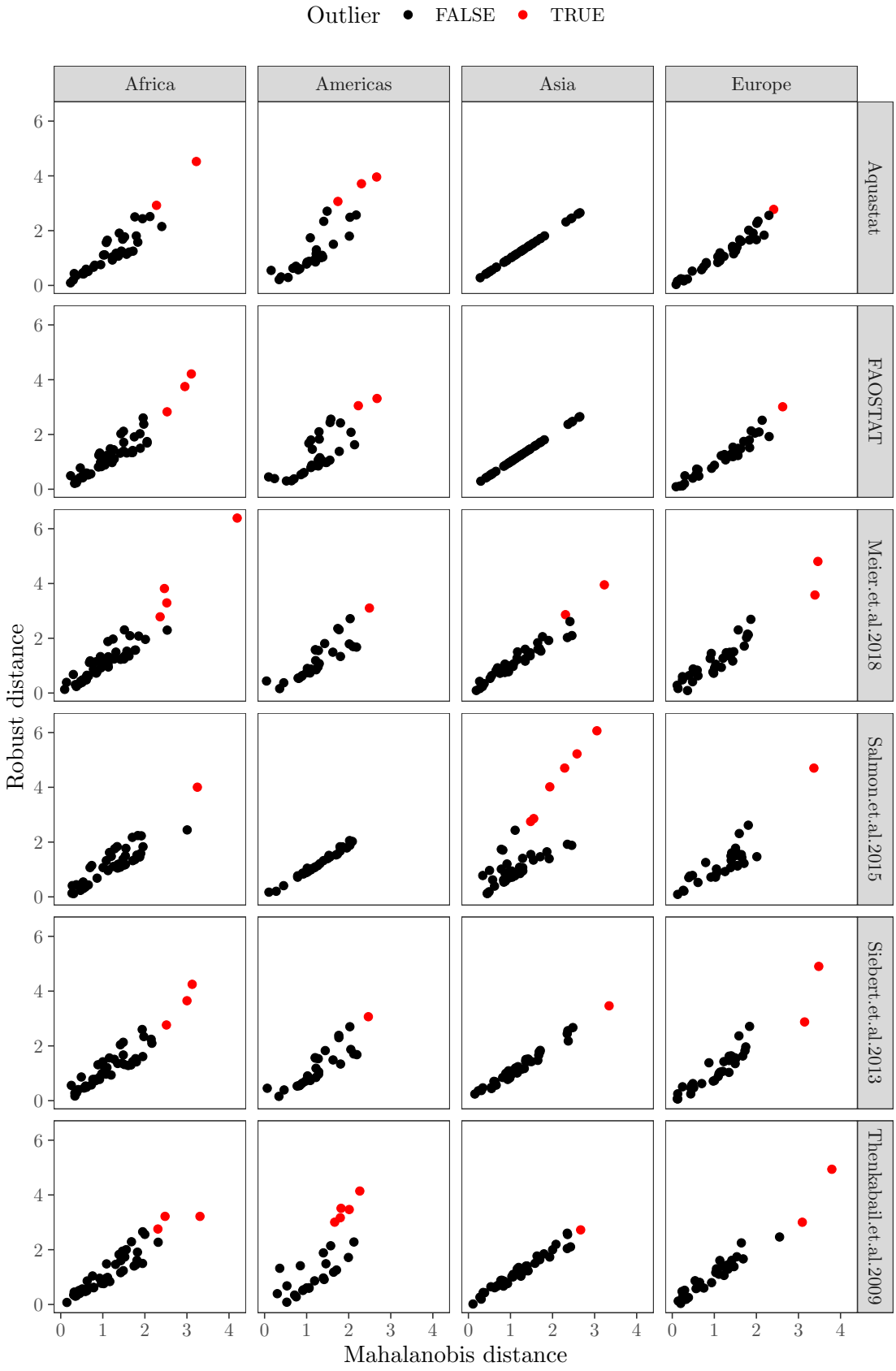


Figure 6: Scatter plot of Mahalanobis vs Robust distances.

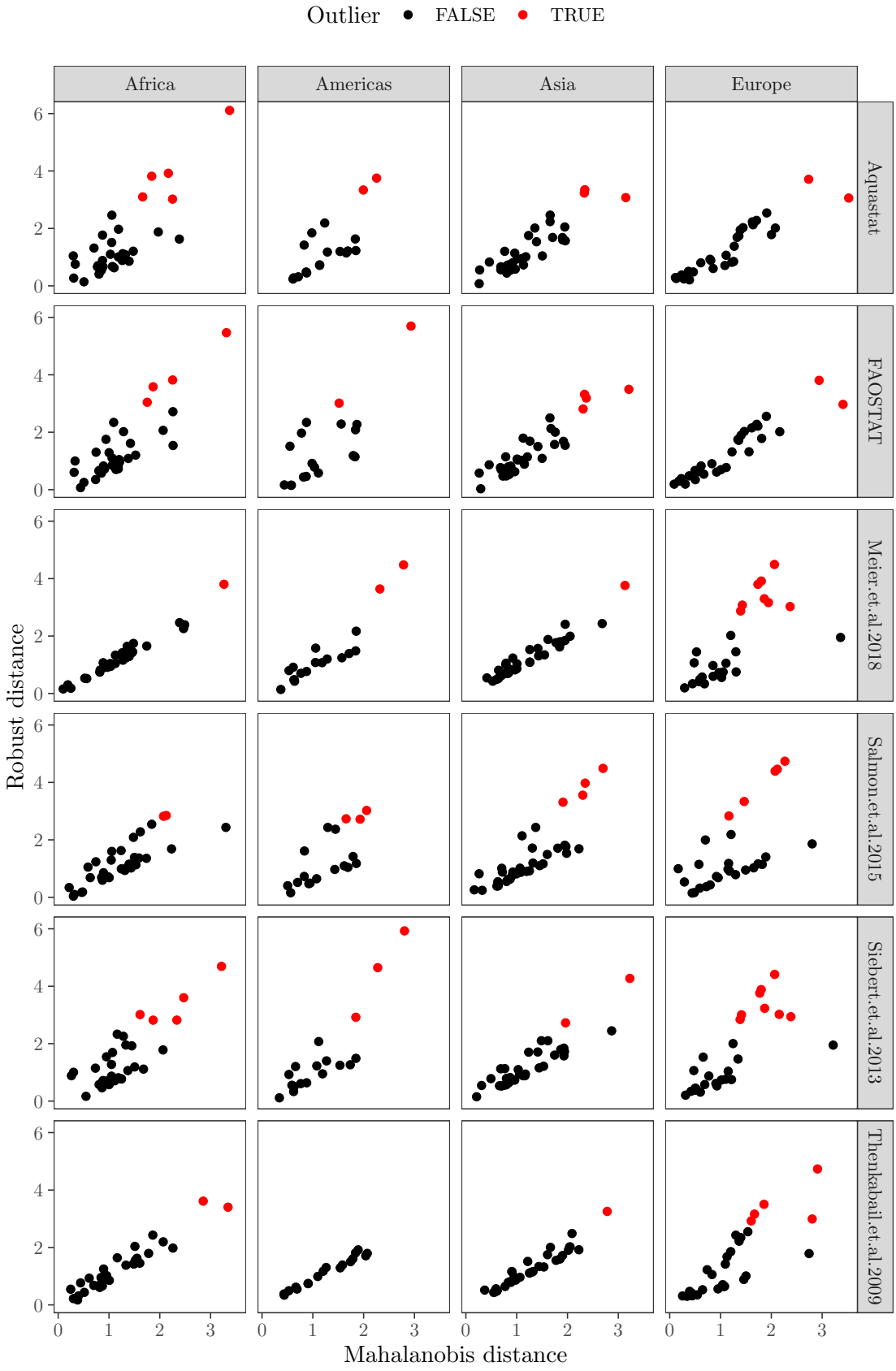


Figure 7: Scatter plot of Mahalanobis vs Robust distances.

5.2 Growth rate between population and irrigated area (β), irrigated area and water required for irrigation (ϕ, δ), and noise (ε)

```
# DEFINE NUMBER OF BOOTSTRAP REPLICAS ----------------------------------------

R <- 5000
```

```
# OLS REGRESSIONS: ALPHA, BETA AND DELTA -------------------------------------

# Create bootstrap function for non-robust OLS
boot.ols <- function(formula, x, i) {
  d <- x[i, ]
  # Bootstrap slope
  fit <-lm(formula, data = d)
  out <-coef(fit)
  return(out)
}
# t1: alpha ols nonrob
# t2: beta ols nonrob

# Bootstrap alpha and beta
dt.regressions <- dt.full %>%
  .[, -2] %>%
  mutate_at(vars(Population, Area.irrigated, Water), funs(log10)) %>%
  data.table()
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
# Regular/Robust regressions
olsB <- dt.regressions[, # Regular OLS regressions
                       .(pop = list(boot(.SD,
                                         statistic = boot.ols,
                                         R = R,
                                         formula = Area.irrigated ~ Population,
                                         parallel = "multicore",
                                         ncpus = floor(detectCores() * 0.75))),
                         water = list(boot(.SD,
                                           statistic = boot.ols,
```

```
                                        R = R,
                                        formula = Water ~ Area.irrigated,
                                        parallel = "multicore",
                                        ncpus = floor(detectCores() * 0.75))),
                        # Robust OLS regressions
                        popR = list(lmrob(Area.irrigated ~ Population)),
                        waterR = list(lmrob(Water ~ Area.irrigated))),
                      by = .(Continent, Dataset)]

# Extract alpha and beta for OLS non robust (population)
ols.nonrobust.pop <- olsB[, "All":= list(lapply(.SD, function(x)
  map(x, "t"))), .SDcols = "pop", .(Continent, Dataset)] %>%
  .[, .("Alpha" = lapply(All, function(x) lapply(x, function(y) y[, 1])),
        "Beta" = lapply(All, function(x) lapply(x, function(y) y[, 2]))),
    .(Continent, Dataset)] %>%
  .[, lapply(.SD, unlist), .SDcols = c("Alpha", "Beta"), .(Continent, Dataset)] %>%
  .[, Regression:= "OLS"] %>%
  .[, Robust:= "NO"]

# Extract phi and delta for OLS non robust (water)
ols.nonrobust.water <- olsB[, "All":= list(lapply(.SD, function(x)
  map(x, "t"))), .SDcols = "water", .(Continent, Dataset)] %>%
  .[, .("Phi" = lapply(All, function(x) lapply(x, function(y) y[, 1])),
        "Delta" = lapply(All, function(x) lapply(x, function(y) y[, 2]))),
    .(Continent, Dataset)] %>%
  .[, lapply(.SD, unlist), .SDcols = c("Phi", "Delta"), .(Continent, Dataset)] %>%
  .[, Regression:= "OLS"] %>%
  .[, Robust:= "NO"]
```

```
# OLS REGRESSIONS ROBUST: ALPHA, BETA AND DELTA ------------------------------

# Create cluster of 4 CPUS to speed up the bootstrapping

# Bootstrap OLS robust
olsR <- olsB[, .(popRob = lapply(popR, function(x) bootcoefs(x,
                                                             R = R,
                                                             method = "frb",
                                                             ncpus = floor(detectCores() * 0.75
                 waterRob = lapply(waterR, function(x) bootcoefs(x,
                                                                 R = R,
                                                                 method = "frb",
                                                                 ncpus = floor(detectCores() *
              .(Continent, Dataset)]

# Extract alpha and beta for OLS robust (population)
ols.robust.pop <- olsR[, "Allpop":= list(lapply(.SD, function(x)
  lapply(x, function(y) y[["bootres"]]))),
```

```
            .SDcols = "popRob", .(Continent, Dataset)] %>%
  .[, "temp":= list(lapply(.SD, function(x)
    lapply(x, function(y) lapply(y, function(z) z[["t"]])))),
    .SDcols = "Allpop", .(Continent, Dataset)] %>%
  .[, .("Alpha" = lapply(temp, function(x)
    lapply(x, function(y) lapply(y, function(z) z[, 1]))),
    "Beta" = lapply(temp, function(x)
      lapply(x, function(y) lapply(y, function(z) z[, 2]))) ,
    .(Continent, Dataset)] %>%
  .[, lapply(.SD, unlist), .SDcols = c("Alpha", "Beta"), .(Continent, Dataset)] %>%
  .[, Regression:= "OLS"] %>%
  .[, Robust:= "YES"]

# Extract phi and delta for OLS robust (water)
ols.robust.water <- olsR[, "Allwater":= list(lapply(.SD, function(x)
  lapply(x, function(y) y[["bootres"]]))),
  .SDcols = "waterRob", .(Continent, Dataset)] %>%
  .[, "temp":= list(lapply(.SD, function(x)
    lapply(x, function(y) lapply(y, function(z) z[["t"]])))),
    .SDcols = "Allwater", .(Continent, Dataset)] %>%
  .[, .("Phi" = lapply(temp, function(x)
    lapply(x, function(y) lapply(y, function(z) z[, 1]))),
    "Delta" = lapply(temp, function(x)
      lapply(x, function(y) lapply(y, function(z) z[, 2]))) ,
    .(Continent, Dataset)] %>%
  .[, lapply(.SD, unlist), .SDcols = c("Phi", "Delta"), .(Continent, Dataset)] %>%
  .[, Regression:= "OLS"] %>%
  .[, Robust:= "YES"]
```

```
# SMA REGRESSIONS: ALPHA AND BETA ------------------------------------------

# Create bootstrap function for robust and non-robust SMA
boot.sma <- function(formula, x, i) {
  d <- x[i, ]
  # Bootstrap coefficients (non-robust)
  fit1 <- sma(formula, data = d, method = "SMA")
  # Bootstrap coefficients (robust)
  fit2 <- sma(formula, data = d, method = "SMA", robust = TRUE)
  coef1 <- coef(fit1)
  coef2 <- coef(fit2)
  all <- c(coef1, coef2)
  return(all)
}
# t1: alpha sma nonrob
# t2: beta sma nonrob
# t3: alpha sma rob
# t4: beta sma rob
```

```
# Bootstrap alpha and beta
smaB <- dt.regressions[, list(list(boot(.SD,
                                        statistic = boot.sma,
                                        R = R,
                                        formula = Area.irrigated ~ Population,
                                        parallel = "multicore",
                                        ncpus = floor(detectCores() * 0.75)))),
                         .(Continent, Dataset)]
```

```
# EXTRACT ALPHA AND BETA ROBUST SMA ----------------------------------------

# Extract alpha and beta SMA (non-robust)
sma.nonrobust.pop <- smaB[, "All":= list(lapply(V1, function(x) x["t"]))] %>%
  .[, list("Alpha" = lapply(All, function(x) lapply(x, function(y) y[, 1])),
           "Beta" = lapply(All, function(x) lapply(x, function(y) y[, 2]))),
      .(Continent, Dataset)] %>%
  .[, lapply(.SD, unlist), .SDcols = c("Alpha", "Beta"), .(Continent, Dataset)] %>%
  .[, Regression:= "SMA"] %>%
  .[, Robust:= "NO"]

# Extract alpha and beta SMA (robust)
sma.robust.pop <- smaB[, "All":= list(lapply(V1, function(x) x["t"]))] %>%
  .[, list("Alpha" = lapply(All, function(x) lapply(x, function(y) y[, 3])),
           "Beta" = lapply(All, function(x) lapply(x, function(y) y[, 4]))),
      .(Continent, Dataset)] %>%
  .[, lapply(.SD, unlist), .SDcols = c("Alpha", "Beta"), .(Continent, Dataset)] %>%
  .[, Regression:= "SMA"] %>%
  .[, Robust:= "YES"]
```

```
fwrite(sma.nonrobust.pop, "sma.nonrobust.pop.csv")
fwrite(sma.robust.pop, "sma.robust.pop.csv")
```

```
# CREATE FINAL DATA SETS WITH ALL BOOTSTRAP SAMPLES ------------------------

# For beta and alpha
boot.samples.pop <- rbind(ols.nonrobust.pop,
                          ols.robust.pop,
                          sma.nonrobust.pop,
                          sma.robust.pop) %>%
  .[order(Continent, Dataset)]

# For delta
boot.samples.water <- rbind(ols.nonrobust.water,
                            ols.robust.water) %>%
  .[order(Continent, Dataset)]
```

```
# EXPORT BOOTSTRAP SAMPLES -------------------------------------------------
```

```r
fwrite(boot.samples.pop, "boot.samples.pop.csv")
fwrite(boot.samples.water, "boot.samples.water.csv")
```

```r
# PREDICT ALPHA FROM BETA ------------------------------------------------

# Predict Alpha from Beta
summary.beta.alpha <- boot.samples.pop %>%
  lm(.$Alpha ~ .$Beta,
     data = .)

# Get intercept, slope and epsilon values
Intercept <- coef(summary.beta.alpha)[[1]]
Slope <- coef(summary.beta.alpha)[[2]]
Epsilon <- summary(summary.beta.alpha)$sigma %>%
  .^2
```

```r
# CREATE THE LOOKUP TABLE ------------------------------------------------

# Create vector to change columns
col_names <- c("Continent", "Dataset", "Regression", "Robust")
col_names2 <- col_names[!col_names %in% "Regression"]

# Create lookup table: population
lookup.pop <- boot.samples.pop[order(Beta), .SD, col_names] %>%
  .[, ID:= 1:.N, col_names] %>%
  .[, index:= paste(Continent, Dataset, Regression, Robust, ID, sep = "_")]

setkey(lookup.pop, index)

# Create lookup table: water
lookup.water <- boot.samples.water[order(Delta), .SD, col_names2] %>%
  .[, ID:= 1:.N, col_names2] %>%
  .[, index:= paste(Continent, Dataset, Robust, ID, sep = "_")]

setkey(lookup.water, index)
```

```r
# EXPORT BOOTSTRAP SAMPLES TO CSV ----------------------------------------

fwrite(lookup.pop, "lookup.pop.csv")
fwrite(lookup.water, "lookup.water.csv")
```

5.3 Population baseline values (N)

```r
# CREATE DATA FRAME WITH POPULATION BASELINE VALUES ----------------------

# Prepare population values between 1999-2015
temp <- df$population %>%
  # Filter out Oceania
```

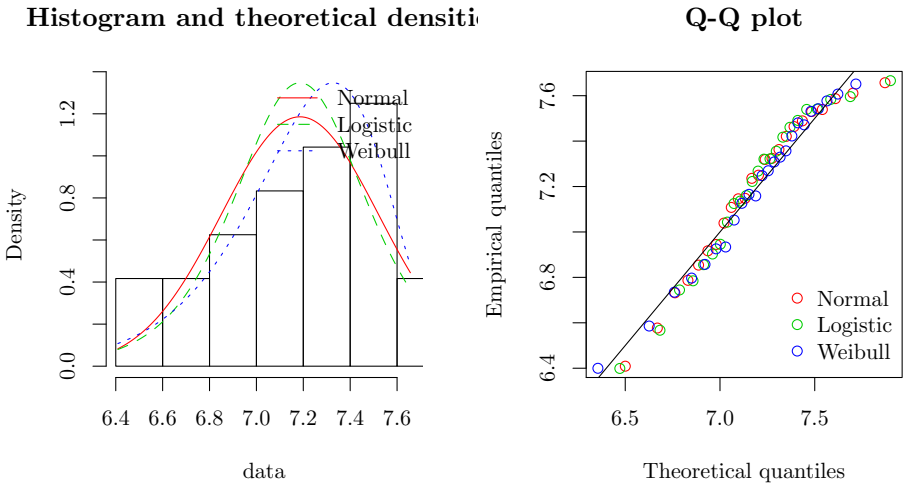
```
  filter(!Continent == "Oceania") %>%
  select(Continent, Codes, Estimate, Year.1999:Year.2012) %>%
  gather(Year, Population, Year.1999:Year.2012) %>%
  mutate(Continent = fct_recode(Continent,
                                "Americas" = "S.America",
                                "Americas" = "N.America")) %>%
  group_by(Continent, Year, Estimate) %>%
  summarise(Population = sum(Population)) %>%
  # Multiply by 1000 to get original population values
  mutate_at(vars(Population), funs(.*10^3)) %>%
  separate(., Year,
           into = c("dummy", "Year")) %>%
  # Drop dummy columns
  mutate_at(vars(Year), funs(as.numeric)) %>%
  mutate(t = 2050 - Year) %>%
  rename(N = Population) %>%
  select(Continent, t, N)

# Create population lookup dataset
population <- setDT(temp)[order(Continent)] %>%
  .[, index:= paste(Continent, t, sep = "_")] %>%
  # To get million population
  .[, N:= N / 10 ^6] %>%
  .[, .(N, index)]

setkey(population, index)
fwrite(population, "population.csv")
```

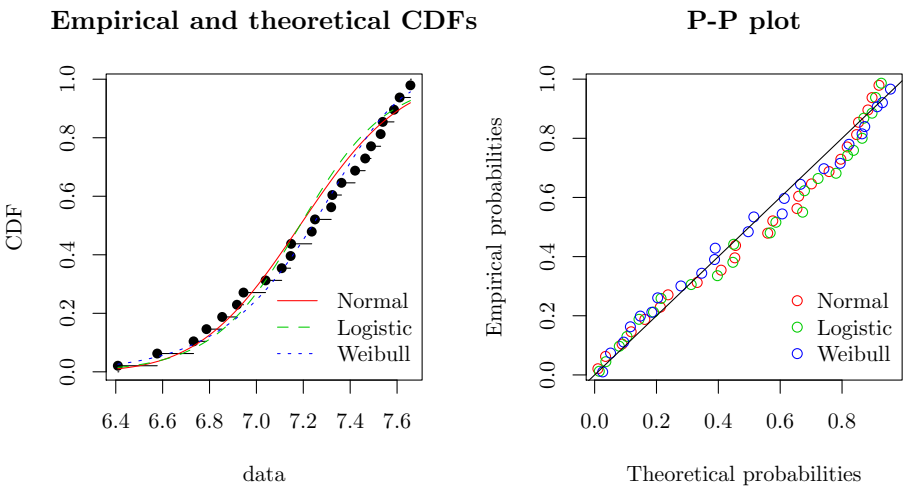
5.4 Population growth rates (r)

```
# DEFINE POPULATION GROWTH RATES DISTRIBUTIONS ----------------------------

# Prepare data frame with growth rates 2015-2050
df.rate1 <- df$growth.rate.estimate %>%
  select(Continent, Estimate, Year.2015.2020:Year.2045.2050) %>%
  gather(Period, Value, Year.2015.2020:Year.2045.2050) %>%
  # Unite population growth rates for N.America and S.America,
  # and consider both regions as one (Americas)
  mutate(Continent = fct_recode(Continent,
                                "Americas" = "S.America",
                                "Americas" = "N.America")) %>%
  # Exclude Oceania
  filter(!Continent == "Oceania")

# Prepare data frame with growth rates 2000-2015
df.rate2 <- df$growth.rate %>%
  select(Continent, Year.2000.2005:Year.2010.2015) %>%
```

```
  gather(Period, Value, Year.2000.2005:Year.2010.2015) %>%
  mutate(Continent = fct_recode(Continent,
                                "Americas" = "S.America",
                                "Americas" = "N.America")) %>%
  # Exclude Oceania
  filter(!Continent == "Oceania")

# Create population growth rates data frame
df.rate <- bind_rows(df.rate1, df.rate2) %>%
  # Add constant (+5) to growth rate values to
  # allow fitting distributions later on
  mutate_at(vars(Value), funs(. + 5)) %>%
  split(., .$Continent, drop = TRUE)


# Describe growth rates of continents with a distribution

# Fit possible distributions according to histograms
distr.norm <- lapply(df.rate, function(x) fitdist(x$Value, "norm"))
distr.logis <- lapply(df.rate, function(x) fitdist(x$Value, "logis",
                                                   method = "mme"))
distr.weib <- lapply(df.rate, function(x) fitdist(x$Value, "weibull"))

# Define function to plot
plotDistr <- function(x,...) {
  funs <- c(denscomp, qqcomp, cdfcomp, ppcomp)
  lapply(funs, function(f) f(list(x,...),
                             legendtext = plot.legend))
}

# Plot distributions and fits (Figures 9-12)
par(mfrow = c(2, 2),
    oma = c(0, 0, 2, 0))

plot.legend <- c("Normal", "Logistic", "Weibull")

for(i in names(distr.norm)) {
  gg <- plotDistr(distr.norm[[i]],
                  distr.logis[[i]],
                  distr.weib[[i]])
  title(names(distr.norm[i]), outer = TRUE)
  print(gg)
}
```

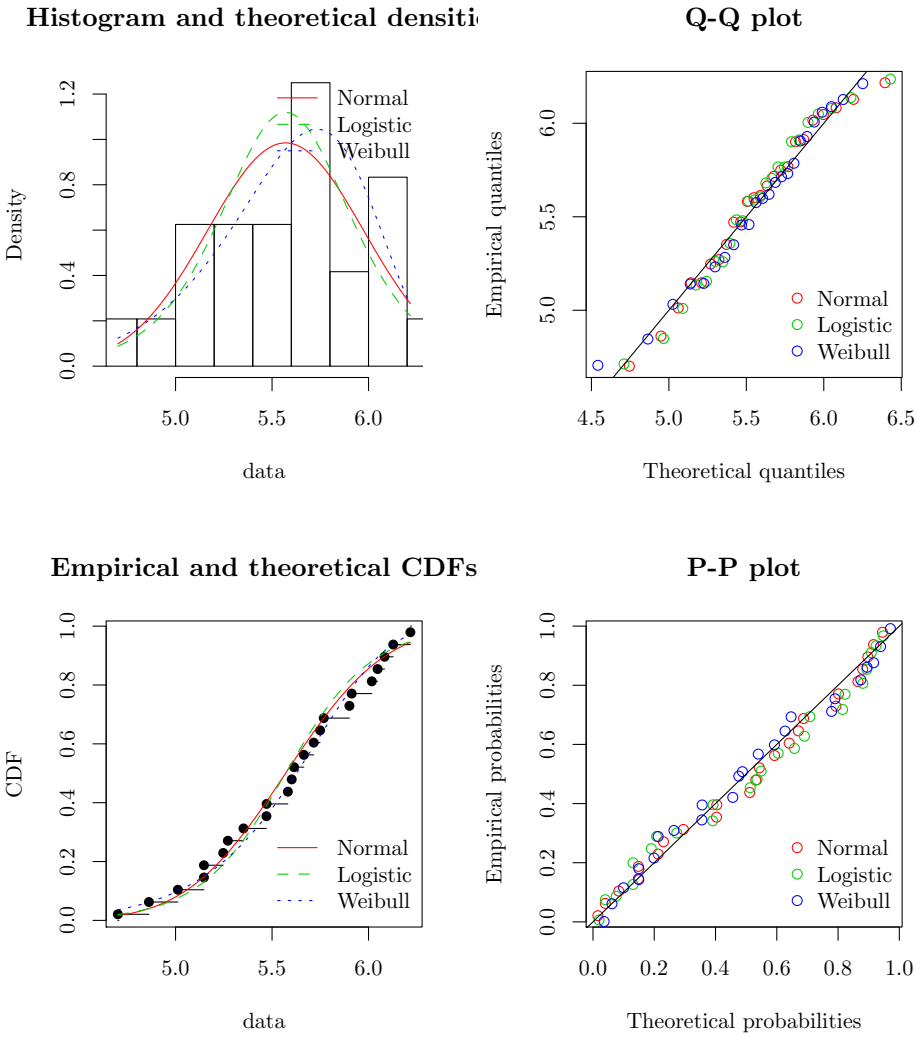
Africa





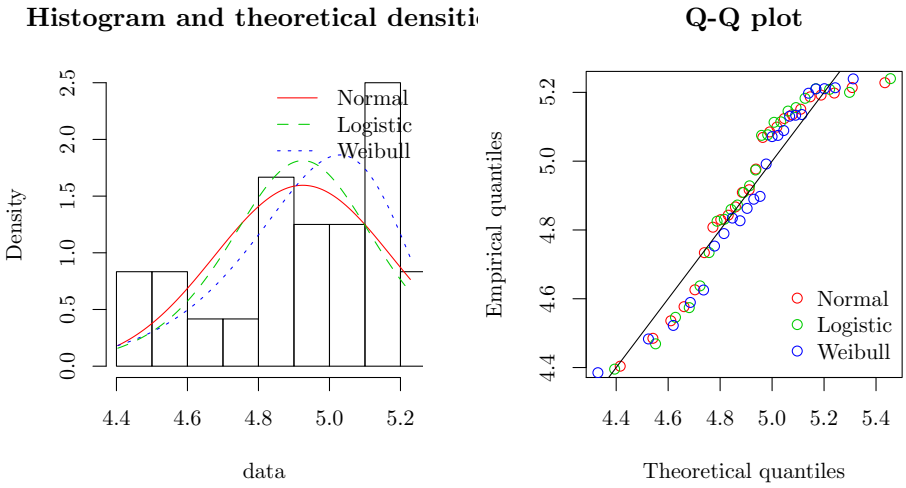

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

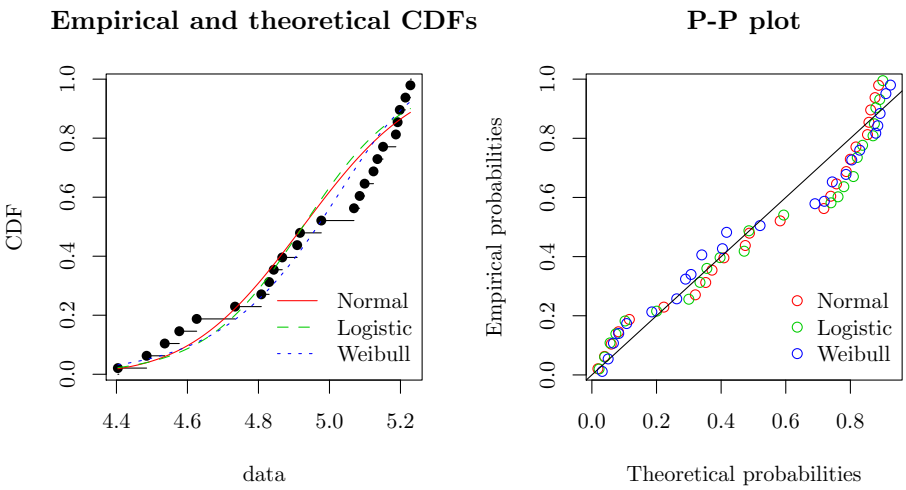
Asia




```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

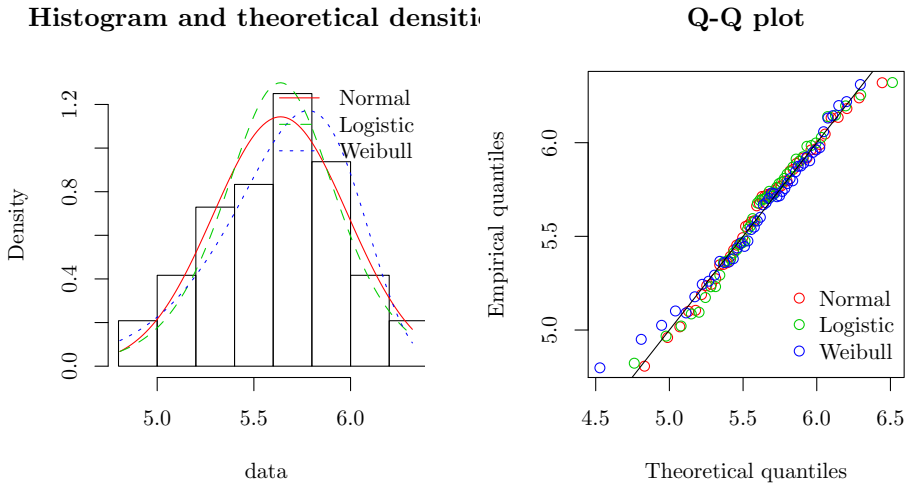
Europe

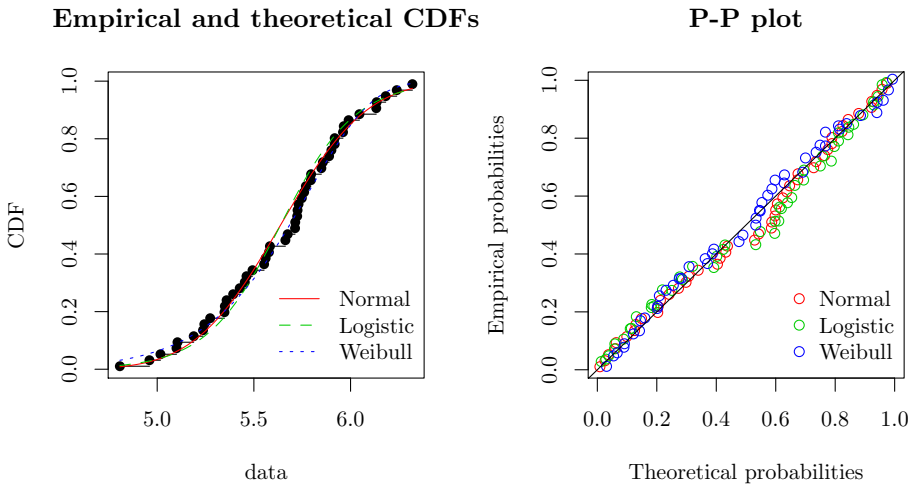





```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

Americas






```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

```
# Create function to assess whether a normal, a logistic or a
# weibull distribution better fits the data
bic.aic <- function(x, y) {
  df <- x %>%
    map(y) %>%
```

```
    data.frame() %>%
    t() %>%
    data.frame() %>%
    rownames_to_column(., var = "Continent")
  return(df)
}

# Assess via BIC
bic.aic(x = distr.norm,
        y = "bic") %>%
  rename(Normal = ".") %>%
  inner_join(., bic.aic(x = distr.logis,
                        y = "bic"),
             by = "Continent") %>%
  rename(Logistic = ".") %>%
  inner_join(., bic.aic(x = distr.weib,
                        y = "bic"),
             by = "Continent") %>%
  rename(Weibull = ".") %>%
  mutate(Model = "BIC")
```

```
##   Continent    Normal  Logistic   Weibull Model
## 1    Africa 22.176234 23.640107 19.532196   BIC
## 2      Asia 31.092193 32.688151 30.187821   BIC
## 3    Europe  7.928691  9.946661  4.671153   BIC
## 4  Americas 42.877046 45.162335 42.951730   BIC
```

```
# The distributions that better fit the data are the following:
# Africa: Weibull distribution
# Asia: Weibull distribution
# Europe: Weibull distribution
# Americas: Normal distribution

# Create data frame with original growth rate values
df.rate.nrm <- bind_rows(df.rate1, df.rate2) %>%
  # Divide growth rates per 100 because it is in percentage
  mutate_at(vars(Value), funs(. / 100)) %>%
  split(., .$Continent, drop = TRUE)


# Fit a normal distribution as we will need the fit for the Americas
distr.norm2 <- lapply(df.rate.nrm, function(x) fitdist(x$Value, "norm"))

# List with the parameters for the distribution of growth rates
growth.rate.distr <- list(distr.weib$Africa$estimate,
                          distr.weib$Asia$estimate,
                          distr.weib$Europe$estimate,
                          distr.norm2$Americas$estimate)
```

```r
# Name the slots of the list
names(growth.rate.distr) <- c("Africa", "Asia", "Europe", "Americas")
```

5.5 Cropland available (K)

```r
# INTEGRATE CROPLAND AVAILABLE WITH MODEL OUTPUT UNCERTAINTY --------------

# Prepare dataset by Zhang: create continental frame
df.zhang <- df$zhang.land.available %>%
  # Transform km2 to ha
  mutate_at(vars(Mkm2, Baseline), funs(.*100)) %>%
  group_by(Continent, Estimation, Baseline) %>%
  summarise(Min = min(Mkm2),
            Max = max(Mkm2)) %>%
  # Conditional mutation: create column filled with
  # either the minimum value (if both projections decrease)
  # or the maximum value (if both projections increase)
  mutate(Value = ifelse(Max < Baseline, Min, Max)) %>%
  data.frame() %>%
  filter(!Continent == "World") %>%
  mutate(Continent = fct_recode(Continent,
                                "Asia" = "China",
                                "Asia" = "India",
                                "Americas" = "US",
                                "Americas" = "S.America",
                                "Europe" = "Russia"))

# Prepare dataset by Zhang: calculate min and max values
temp <- df.zhang %>%
  select(Continent, Estimation, Min, Max) %>%
  split(.,list(.$Continent, .$Estimation),
        drop = TRUE) %>%
  lapply(., function(x)  {
    x$min <- sum(x$Min)
    x$max <- sum(x$Max)
    return(x[1, 5:6])
  })

# Create final data set by Zhang to plot
cropland.1 <- temp %>%
  map(data.frame) %>%
  rbindlist(., idcol = "Continent") %>%
  separate(., col = Continent,
           into = c("Continent", "Estimation"))

cropland <- cropland.1 %>%
```

```
  gather(Parameter, Value, min:max) %>%
  group_by(Continent) %>%
  summarise(min = min(Value),
            max = max(Value)) %>%
  split(., .$Continent)
```

5.6 Water available (W_a)

```
# DEFINE DISTRIBUTIONS FOR THE TOTAL WATER AVAILABLE -------------------------

# Read in dataset (# 10^9 m3/yr (It is already in km3)
water.availability <- fread("aquastat_water.csv",
                            nrows = 182)[, .(Country, Value)]

# Get the codes and the continents
addAll(water.availability, "Aquastat")
```

```
## Warning in countrycode(x, origin = "country.name", destination = "un"): Some values were not
```

```
## Warning in countrycode(x, origin = "country.name", destination = "continent"): Some values w
```

```
# Compute 20% uncertainty
water.availability.dt <- water.availability[!Continent == "Oceania"] %>%
  .[, sum(Value, na.rm = TRUE), Continent] %>%
  .[, uncertainty:= round(V1 * 0.20, digits = 0)] %>%
  .[, lower:= round(V1 - uncertainty, digits = 0)] %>%
  .[, upper:= round(V1 + uncertainty, digits = 0)] %>%
  split(., .$Continent)
```

6 Creation of the sample matrix

```
# CREATE THE SAMPLE MATRIX ---------------------------------------------------

# Create a vector with the name of the columns
parameters <- c("X1", "X2", "X3", "X4", "W1", "W3", "W4", "r",
                "gamma", "Y0", "epsilon", "t", "K", "W_a", "eta")

# Obtain number of parameters
k <- length(parameters)

# Select sample size
n <- 2 ^ 14

# Create vector with the continents
Continents <- c("Africa", "Americas", "Asia", "Europe")

# Create an A, B and AB matrices for each continent
```

```
AB <- lapply(Continents, function(Continents)
  sobol_matrices(n = n, k = k) %>%
               data.table())

# Name the slots, each is a continent
names(AB) <- Continents

# Name the columns
AB <- lapply(AB, setnames, parameters)
```

```
# CREATE THE SAMPLE MATRICES FOR THE CLUSTERED PARAMETERS ---------------------

# Create vectors with the name of the parameters within each cluster
irrigation <- match(c("X1", "YO", "W1", "W_a", "eta"), parameters)
population2 <- match(c("r", "gamma"), parameters)
model <- match(c("X2", "X3", "X4", "W3", "W4", "epsilon"), parameters)


# Create an A, B and AB matrices for the clustered parameters; retrieve
# only the AB

AB.cluster <- lapply(Continents, function(Continents)
  sobol_matrices(n = n,
                 k = k,
                 cluster = list(irrigation, population2, model))) %>%
  lapply(., function(x) x[((2*n) + 1):nrow(x), ]) %>%
  lapply(., data.table)

# Name the slots, each is a continent
names(AB.cluster) <- Continents

# Name the columns
AB.cluster <- lapply(AB.cluster, setnames, parameters)

# Merge the sample matrix and the sample matrix of the
# clustered parameters

for(i in names(AB)) {
  AB[[i]] <- rbind(AB[[i]], AB.cluster[[i]])
}
```

```
# CHECK NUMBER OF BOOTSTRAP SAMPLES OF BETA, DELTA, ETC. ----------------------

N.boot <- boot.samples.pop[, .(N = .N),
                           .(Continent, Dataset, Regression, Robust)] %>%
  .[, N] %>%
  .[1]
```

```r
print(N.boot)
```

```
## [1] 5000
```

```r
# TRANSFORM THE SAMPLE MATRIX ---------------------------------------------

# Create function to transform the parameters that
# have the same distribution in all continents
transform.sobol <- function(X) {
  X[, X1:= floor(X1 * (6-1+1)) + 1][, X1:= ifelse(X1 == 1, "Aquastat",
                                                  ifelse(X1 == 2, "FAOSTAT",
                                                         ifelse(X1 == 3, "Siebert.et.al.2013",
                                                                ifelse(X1 == 4, "Meier.et.al.2
                                                                       ifelse(X1 == 5, "Salmon
                                                                              "Thenkabail.et.a
  X[, X2:= floor(X2 * (2-1+1)) + 1][, X2:= ifelse(X2==1, "OLS", "SMA")]
  X[, X3:= floor(X3 * (2-1+1)) + 1][, X3:= ifelse(X3==1, "YES", "NO")]
  X[, X4:= floor(X4 * (N.boot - 1)) + 1]
  X[, W1:= floor(W1 * (6-1+1)) + 1][, W1:= ifelse(W1 == 1, "Aquastat",
                                                  ifelse(W1 == 2, "FAOSTAT",
                                                         ifelse(W1 == 3, "Siebert.et.al.2013",
                                                                ifelse(W1 == 4, "Meier.et.al.2
                                                                       ifelse(W1 == 5, "Salmon
                                                                              "Thenkabail.et.a
  X[, W3:= floor(W3 * (2-1+1)) + 1][, W3:= ifelse(W3==1, "YES", "NO")]
  X[, W4:= floor(W4 * (N.boot - 1)) + 1]
  X[, gamma:= 0.02 * qnorm(gamma) + 1]
  X[, epsilon:= Epsilon * qnorm(epsilon) + 0]
  X[, t:= floor(t * (51-38 + 1)) + 38]
  X[, eta:= qunif(eta, min = 0.2, max = 0.5)]
  return(X)
}

AB <- lapply(AB, transform.sobol)

# Transform the parameters with their appropriate distributions
transform.sobol.continents <- function(AB) {
  for(i in names(AB)) {
    if(i == "Africa") {
      # Weibull distribution, substract the constant and divide by 100
      # because original values were in percentage
      AB[[i]][, r:= (growth.rate.distr$Africa[[2]] *
                       (-log(1 - r)) ^ (1/growth.rate.distr$Africa[[1]])
                     -5) / 100]
      # Uniform distribution
      AB[[i]][, Y0:= Y0 *
                (total.area.irrigated$Africa$max-total.area.irrigated$Africa$min) +
                total.area.irrigated$Africa$min]
```

```
    # Unifrom distribution
    AB[[i]][, K:= K * (cropland$Africa$max-cropland$Africa$min) +
              cropland$Africa$min]
    # Uniform distribution
    AB[[i]][, W_a:= qunif(W_a, min = water.availability.dt$Africa$lower,
                          max = water.availability.dt$Africa$upper)]
  }
  if(i == "Asia") {
    # Weibull distribution, substract the constant and divide by 100
    # because original values were in percentage
    AB[[i]][, r:= (growth.rate.distr$Asia[[2]] *
                     (-log(1 - r)) ^ (1/growth.rate.distr$Asia[[1]])
                   -5) / 100]
    # Uniform distribution
    AB[[i]][, Y0:= Y0  *
              (total.area.irrigated$Asia$max-total.area.irrigated$Asia$min) +
              total.area.irrigated$Asia$min]
    # Unifrom distribution
    AB[[i]][, K:= K * (cropland$Asia$max-cropland$Asia$min) +
              cropland$Asia$min]
    # Uniform distribution
    AB[[i]][, W_a:= qunif(W_a, min = water.availability.dt$Asia$lower,
                          max = water.availability.dt$Asia$upper)]
  }
  if(i == "Americas") {
    # Normal distribution
    AB[[i]][, r:= (growth.rate.distr$Americas[[2]] *
                     qnorm(r) + growth.rate.distr$Americas[[1]])]
    # Uniform distribution
    AB[[i]][, Y0:= Y0  *
              (total.area.irrigated$Americas$max-total.area.irrigated$Americas$min) +
              total.area.irrigated$Americas$min]
    # Unifrom distribution
    AB[[i]][, K:= K * (cropland$Americas$max-cropland$Americas$min) +
              cropland$Americas$min]
    # Uniform distribution
    AB[[i]][, W_a:= qunif(W_a, min = water.availability.dt$Americas$lower,
                          max = water.availability.dt$Americas$upper)]
  }
  if(i == "Europe") {
    # Weibull distribution, substract the constant and divide by 100
    # because original values were in percentage
    AB[[i]][, r:=(growth.rate.distr$Europe[[2]] *
                    (-log(1 - r)) ^ (1/growth.rate.distr$Europe[[1]])
                  -5) / 100]
    # Uniform distribution
    AB[[i]][, Y0:= Y0 *
```

```
                (total.area.irrigated$Europe$max-total.area.irrigated$Europe$min) +
                total.area.irrigated$Europe$min]
      # Unifrom distribution
      AB[[i]][, K:= K * (cropland$Europe$max-cropland$Europe$min) +
                cropland$Europe$min]
      # Uniform distribution
      AB[[i]][, W_a:= qunif(W_a, min = water.availability.dt$Europe$lower,
                            max = water.availability.dt$Europe$upper)]
    }
  }
  return(AB)
}

AB <- transform.sobol.continents(AB)
```

```
# WRITE FINAL DATA TABLE ----------------------------------------------------

final.dt <- rbindlist(AB, idcol = "Continent")
```

```
# EXPORT FINAL DATA TABLE ---------------------------------------------------

fwrite(final.dt, "final.dt.csv")
print(final.dt)
```

```
##          Continent                    X1  X2  X3   X4                    W1  W3
##       1:    Africa      Meier.et.al.2018 SMA  NO 2500      Meier.et.al.2018  NO
##       2:    Africa     Salmon.et.al.2015 OLS  NO 1250     Salmon.et.al.2015 YES
##       3:    Africa               FAOSTAT SMA YES 3750               FAOSTAT  NO
##       4:    Africa    Siebert.et.al.2013 OLS  NO  625 Thenkabail.et.al.2009  NO
##       5:    Africa Thenkabail.et.al.2009 SMA YES 3125    Siebert.et.al.2013 YES
##      ---
## 1310716:    Europe               FAOSTAT SMA YES 4962     Salmon.et.al.2015  NO
## 1310717:    Europe     Salmon.et.al.2015 OLS  NO 2462               FAOSTAT YES
## 1310718:    Europe      Meier.et.al.2018 OLS YES 1213    Siebert.et.al.2013 YES
## 1310719:    Europe              Aquastat SMA  NO 3712 Thenkabail.et.al.2009  NO
## 1310720:    Europe              Aquastat SMA  NO 3664    Siebert.et.al.2013  NO
##            W4             r     gamma       Y0      epsilon  t         K
##       1: 2500  0.0223856397 1.0000000 12.45665  0.000000000 45  959.5000
##       2: 3750  0.0200542303 0.9865102 14.41732 -0.100912218 48  836.7500
##       3: 1250  0.0242778499 1.0134898 10.49597  0.100912218 41 1082.2500
##       4:  625  0.0233272313 0.9769930 15.39766 -0.047672489 46  775.3750
##       5: 3125  0.0180808545 1.0063728 11.47631  0.172106852 39 1020.8750
##      ---
## 1310716:   60 -0.0004517236 1.0009826 32.03453 -0.102614192 43  490.7327
## 1310717: 2559  0.0031712483 0.9587551 22.69423  0.099223204 50  604.2327
## 1310718: 1310 -0.0021985853 1.0147501 27.36438  0.402061601 47  547.4827
## 1310719: 3809  0.0009159681 0.9877191 36.70468 -0.001350505 40  660.9827
## 1310720: 3224 -0.0002631836 0.9941865 38.05864 -0.057671016 41  598.8916
```

```
##               W_a       eta
##       1: 3928.000 0.3500000
##       2: 4321.000 0.2750000
##       3: 3535.000 0.4250000
##       4: 3731.500 0.3875000
##       5: 4517.500 0.2375000
##      ---
## 1310716: 6740.536 0.4739075
## 1310717: 5425.536 0.3239075
## 1310718: 6083.036 0.3989075
## 1310719: 7398.036 0.2489075
## 1310720: 6000.928 0.3946136
```

7 The model

```
# DEFINE THE MODEL ------------------------------------------------------------

model <- function(X) {
  # Extract beta
  Beta <- lookup.pop[.(paste0(X[, 1:5], collapse = "_"))][, Beta]
  # Select population
  N <- population[.(paste0(X[, c("Continent", "t")], collapse = "_"))] %>%
    .[, N]
  # Extract phi and delta
  tmp <- lookup.water[.(paste0(X[, c(1, 6:8)], collapse = "_"))]
  Phi <- tmp[, Phi]
  Delta <- tmp[, Delta]
  # COMPUTE THE MODEL ---------------------------
  Y <- X[, Y0] + 10 ^ (Intercept + Slope * Beta + X[, epsilon]) *
  ((N ^ (1- X[, gamma]) + X[, r] * X[, t] * (1 - X[, gamma])) ^
     (Beta / (1 - X[, gamma])) - N ^ Beta)
  # Compute how much water will we need to irrigate Y
  w <- (10 ^ Phi) * Y ^ Delta
  # Compute how much water we have
  # available for irrigation --------------------
  w_i <- X[, W_a] * X[, eta]
  # Compute the total extension that can
  # be irrigated with the water
  # we have available for irrigation ------------
  Y.max <- (w_i / 10 ^ Phi) ^ (1 / Delta)
  # Constrain
  if(Y > X[, K]) {
    Y <- X[, K]
  }
  if(Y < 0) {
    Y <- X[, Y0]
  }
```

```
  if(Y > Y.max) {
    Y <- Y.max
    if(Y < X[, YO]) {
      Y <- X[, YO]
    }
  }
  return(c(Beta, N, Phi, Delta, w, w_i, Y.max, Y))
}
```

```
# RUN MODEL USING PARALLEL COMPUTING -------------------------------------

# Define parallel computing
cl <- makeCluster(floor(detectCores() * 0.75))
registerDoParallel(cl)

# Run model in parallel
Y <- foreach(i=1:nrow(final.dt),
             .packages = c("dplyr", "data.table")) %dopar%
             {
               model(final.dt[i])
             }

# Stop parallel cluster
stopCluster(cl)
```

```
# ADD MODEL OUTPUT -------------------------------------------------------

model.output <- c("Beta", "N", "Phi", "Delta", "w", "w_i", "Y.max", "Y")

full.dt <- cbind(final.dt, data.table(do.call(rbind, Y))) %>%
  setnames(., paste("V", 1:length(model.output), sep = ""), model.output)

# Select the A and B matrix only (for uncertainty analysis)
AB.dt <- full.dt[, .SD[1:(n * 2)], Continent]
```

```
# EXPORT MODEL OUTPUT ----------------------------------------------------

fwrite(full.dt, "full.dt.csv")
```

```
# EXPORT AB MATRICES -----------------------------------------------------

fwrite(AB.dt, "AB.dt.csv")
```

8 Uncertainty analysis

```
# COMPUTE QUANTILES ------------------------------------------------------

# Check number and proportion of negative model output values
```

```
AB.dt[, .(negative.runs = sum(Y < 0),
          proportion = sum(Y < 0) / .N),
      Continent]
```

```
##    Continent negative.runs proportion
## 1:    Africa             0          0
## 2:  Americas             0          0
## 3:      Asia             0          0
## 4:    Europe             0          0
```

```
# Compute quantiles
quant <- AB.dt[Y > 0] %>%
  group_by(Continent) %>%
  do(data.frame(t(quantile(.$Y,
                           probs = c(0.005, 0.01, 0.025, 0.975, 0.99, 0.995, 1),
                           na.rm = TRUE))))

# Print the quantiles
print(quant)
```

```
## # A tibble: 4 x 8
## # Groups:   Continent [4]
##   Continent  X0.5.    X1.  X2.5. X97.5.  X99. X99.5. X100.
##   <chr>      <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl> <dbl>
## 1 Africa     15.4   17.3   20.5   780.   945.  1049. 1204.
## 2 Americas   42.6   45.4   48.2   159.   212.   255. 1186.
## 3 Asia      132.   167.   201.    716.   797.   833.  889.
## 4 Europe      1.52   2.85   6.85   77.6  125.   170.  660.
```

```
# COMPUTE QUANTILES AT THE GLOBAL LEVEL ---------------------------------------

projections <- df$projections %>%
  select(Study, `2050`, Group) %>%
  na.omit() %>%
  data.table()

projections <- rbind(projections,
                     data.table(Study = "Rosegrant.et.al.2002",
                                `2050` = 237,
                                Group = 11))

# Assess uncertainty at the global level
global.uncertainty <- AB.dt %>%
  .[, .(Y, Continent)] %>%
  split(., .$Continent) %>%
  lapply(., function(x) x[, Y]) %>%
  do.call("cbind", .) %>%
  data.table() %>%
  .[, Total:= rowSums(.)]
```

```r
# Calculate the 2.5 and the 97.5 quantiles
global.quantile <- quantile(global.uncertainty$Total,
                            probs = c(0.005, 0.01, 0.025, 0.975, 0.99, 0.995, 1),
                            na.rm = TRUE) %>%
  t() %>%
  data.frame()

# Print quantiles of the global uncertainty
print(global.quantile)
```

```
##      X0.5.      X1.    X2.5.   X97.5.     X99.   X99.5.    X100.
## 1 270.0776 293.1642 319.7956 1462.419 1751.077 1961.623 2918.227
```

```r
# READ PROJECTIONS OF IRRIGATED AREAS AT THE CONTINENTAL LEVEL ------------------

irrigated_area_2050 <- fread("irrigated_area_2050.csv")[, World:= NULL]
irrigated_area_2050_dt <- melt(irrigated_area_2050,
                               measure.vars = c("Africa", "Americas", "Europe", "Asia"),
                               variable.name = "Continent")
```

```
## Warning in melt.data.table(irrigated_area_2050, measure.vars = c("Africa", :
## 'measure.vars' [Africa, Americas, Europe, Asia] are not all of the same type. By
## order of hierarchy, the molten data value column will be of type 'double'. All
## measure variables not of type 'double' will be coerced too. Check DETAILS in ?
## melt.data.table for more on coercion.
```

```r
# Get minimum and maximum
prove <- irrigated_area_2050_dt[, .(min = min(value, na.rm = TRUE),
                                    max = max(value, na.rm = TRUE)),
                                Continent]

# Proportion of model runs covered by current projections (min and max)
sapply(Continents, function(x) AB.dt[Continent == x,
                                     sum(Y >= prove[Continent == x, min] &
                                           Y<= prove[Continent == x, max]) / .N])
```

```
##    Africa  Americas      Asia    Europe
## 0.1432495 0.5704956 0.2822876 0.6549377
```

```r
# Proportion of model runs larger than the maximum value projected
sapply(Continents, function(x) AB.dt[Continent == x,
                                     sum(Y >= prove[Continent == x, max]) / .N])
```

```
##    Africa  Americas      Asia    Europe
## 0.8561707 0.4252625 0.7086182 0.2055054
```

```r
# PLOT UNCERTAINTY ----------------------------------------------------------

# Continental level
a <- AB.dt %>%
```

```
  .[Y > 10^0] %>%
  ggplot(., aes(Y)) +
  geom_rect(data = quant,
            aes(xmin = X2.5.,
                xmax = X97.5.,
                ymin = -Inf,
                ymax = Inf,
                group = Continent),
            fill = "green",
            color = "white",
            alpha = 0.1,
            inherit.aes = FALSE) +
  geom_rect(data = cropland.1,
            aes(xmin = min,
                xmax = max,
                ymin = -Inf,
                ymax = Inf,
                group = Continent,
                fill = Estimation),
            color = "black",
            alpha = 0.7,
            inherit.aes = FALSE) +
  scale_fill_manual(guide = FALSE,
                    values = c("chocolate4", "chocolate1")) +
  geom_histogram() +
  geom_vline(data = irrigated_area_2050_dt,
             aes(xintercept = value,
                 colour = Study),
             lty = 2,
             size = 1) +
  labs(x = "Area irrigated 2050 (Mha)",
       y = "Counts") +
  facet_wrap(~Continent,
             ncol = 1,
             scales = "free_y") +
  scale_x_log10(labels = trans_format("log10", math_format(10^.x))) +
  scale_y_continuous(breaks = pretty_breaks(n = 2)) +
  theme_bw() +
  theme(legend.position = "none",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.background = element_rect(fill = "transparent",
                                         color = NA),
        legend.key = element_rect(fill = "transparent",
                                  color = NA))

# Global level
```

```r
# Global level
b <- global.uncertainty %>%
  ggplot(., aes(Total)) +
  geom_rect(data = global.quantile,
            aes(xmin = X2.5.,
                xmax = X97.5.,
                ymin = -Inf,
                ymax = Inf),
            fill = "green",
            color = "white",
            alpha = 0.1,
            inherit.aes = FALSE) +
  geom_rect(data = cropland.1[, .(maximum = sum(max),
                                  minimum = sum(min)), Estimation],
            aes(xmin = minimum,
                xmax = maximum,
                ymin = -Inf,
                ymax = Inf,
                fill = Estimation),
            color = "black",
            alpha = 0.7,
            inherit.aes = FALSE) +
  scale_fill_manual(guide = FALSE,
                    values = c("chocolate4", "chocolate1")) +
  geom_histogram() +
  geom_vline(data = projections[!Study == "Alcamo.et.al.2005"],
             aes(xintercept = `2050`,
                 colour = Study),
             lty = 2,
             size = 1) +
  scale_color_discrete(labels = c("Alcamo et al. 2005",
                                  "Alexandratos and \n Bruinsma 2012",
                                  "Fischer et al. 2007",
                                  "Molden 2007",
                                  "Rosegrant et al. 2002")) +
  labs(x = "Area irrigated 2050 (Mha)",
       y = "",
       color = "Projection") +
  scale_x_log10(labels = trans_format("log10", math_format(10^.x)),
                breaks = trans_breaks("log10", function(x) 10^x),
                # Limit the x axis for better visualization
                limits = c(200, 4300)) +
  theme_bw() +
  theme(legend.position = c(0.82, 0.58),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.background = element_rect(fill = alpha("white", 0.7)),
```

```
        legend.key = element_rect(fill = "transparent",
                                  color = NA))
```

```
plot_grid(a + labs(x = "", y = "Counts"),
          b + labs(x = "Area irrigated 2050 (Mha)", y = "Counts"),
          ncol = 1,
          labels = "AUTO",
          align = "hv",
          rel_heights = c(1, 0.6))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

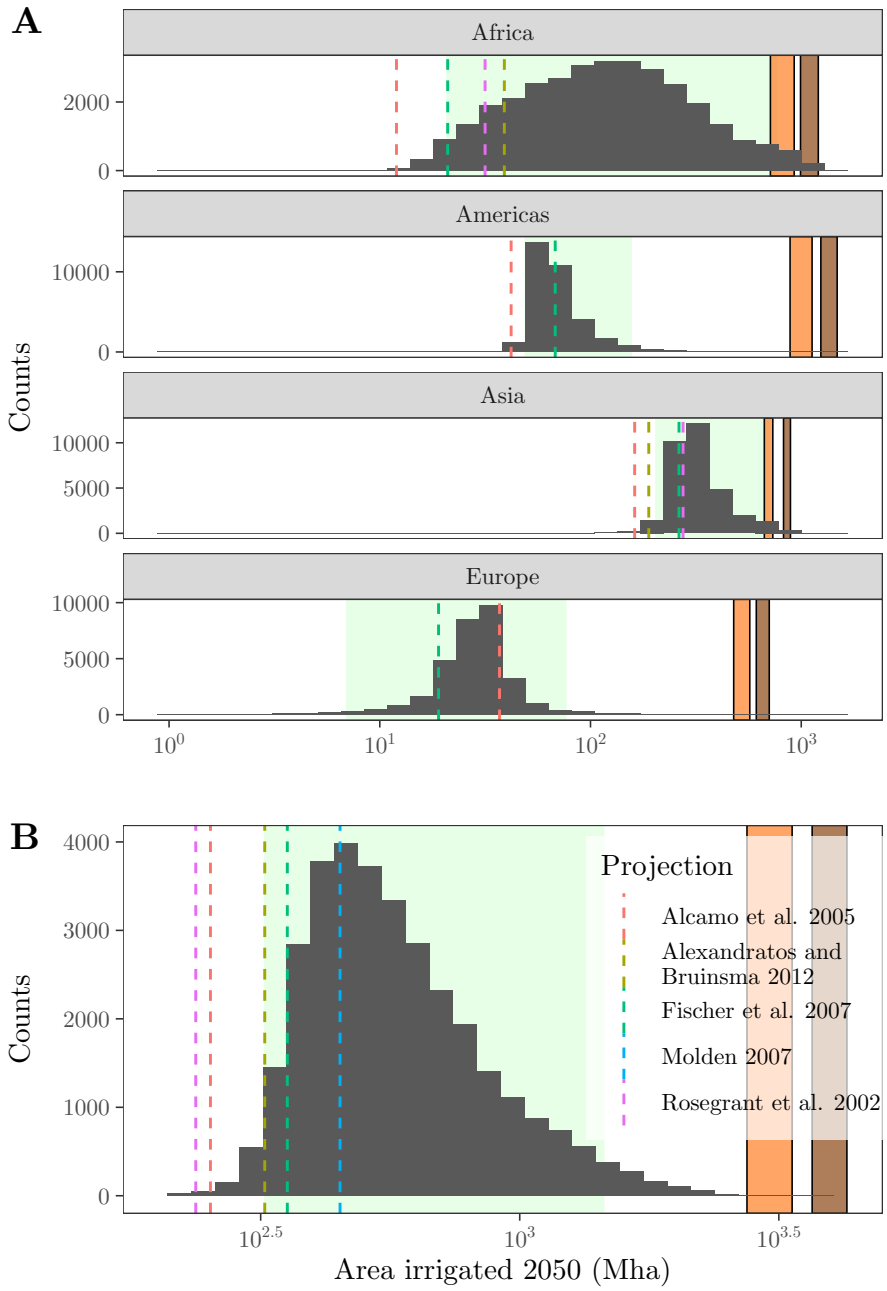
```
## Warning: Removed 8 rows containing missing values (geom_vline).
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 33 rows containing non-finite values (stat_bin).
```

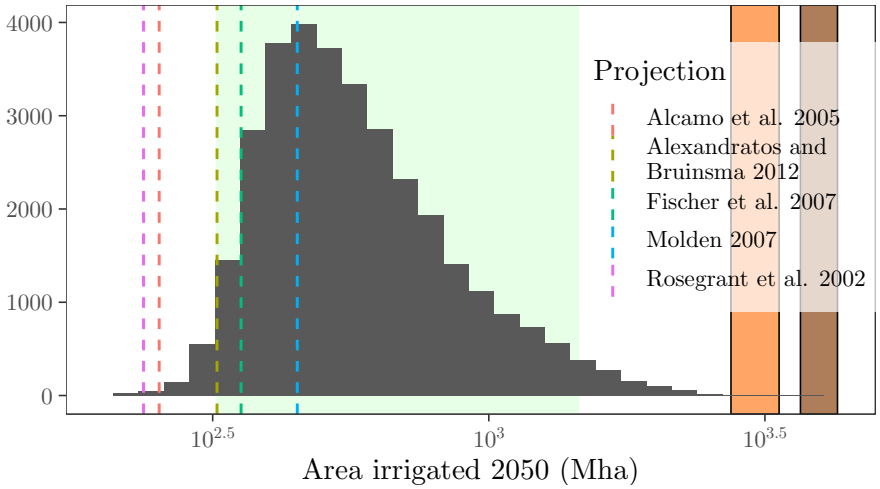
```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

```
## Warning: Graphs cannot be horizontally aligned unless the axis parameter is set.
## Placing graphs unaligned.
```




```
b + theme(plot.margin = margin(l = 0, unit = "cm"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 33 rows containing non-finite values (stat_bin).
## Warning: Removed 2 rows containing missing values (geom_bar).
```




```
# EXPORT GLOBAL UNCERTAINTY ------------------------------------------------

fwrite(global.uncertainty, "global.uncertainty.csv")
```

```
# UNCERTAINTY STATISTICS ---------------------------------------------------

# How many model runs hit K?
AB.dt[, sum((Y == K) / .N) * 100, Continent]
```

```
##    Continent          V1
## 1:    Africa 2.066040039
## 2:  Americas 0.003051758
## 3:      Asia 2.624511719
## 4:    Europe 0.027465820
```

```
# Compare the global projection to previous projections
global.uncertainty[, .(N = .N,
                       larger.Alcamo.2007 = sum((Total > 262) / .N) * 100,
                       larger.FAO = sum((Total > 322) / .N) * 100,
                       larger.Molden = sum((Total > 450) / .N) * 100,
                       much.larger.Molden = sum((Total > 675) / .N) * 100,
                       twice.Molden = sum((Total > 900) / .N) * 100,
                       three.Molden = sum((Total > 450 * 3) / .N) * 100,
                       less.100 = sum((Total < 10^2) / .N) * 100)]
```

```
##        N larger.Alcamo.2007 larger.FAO larger.Molden much.larger.Molden
## 1: 32768           99.61243   97.32361      69.72656           29.82788
##    twice.Molden three.Molden    less.100
## 1:     13.85498     3.646851 0.006103516
```

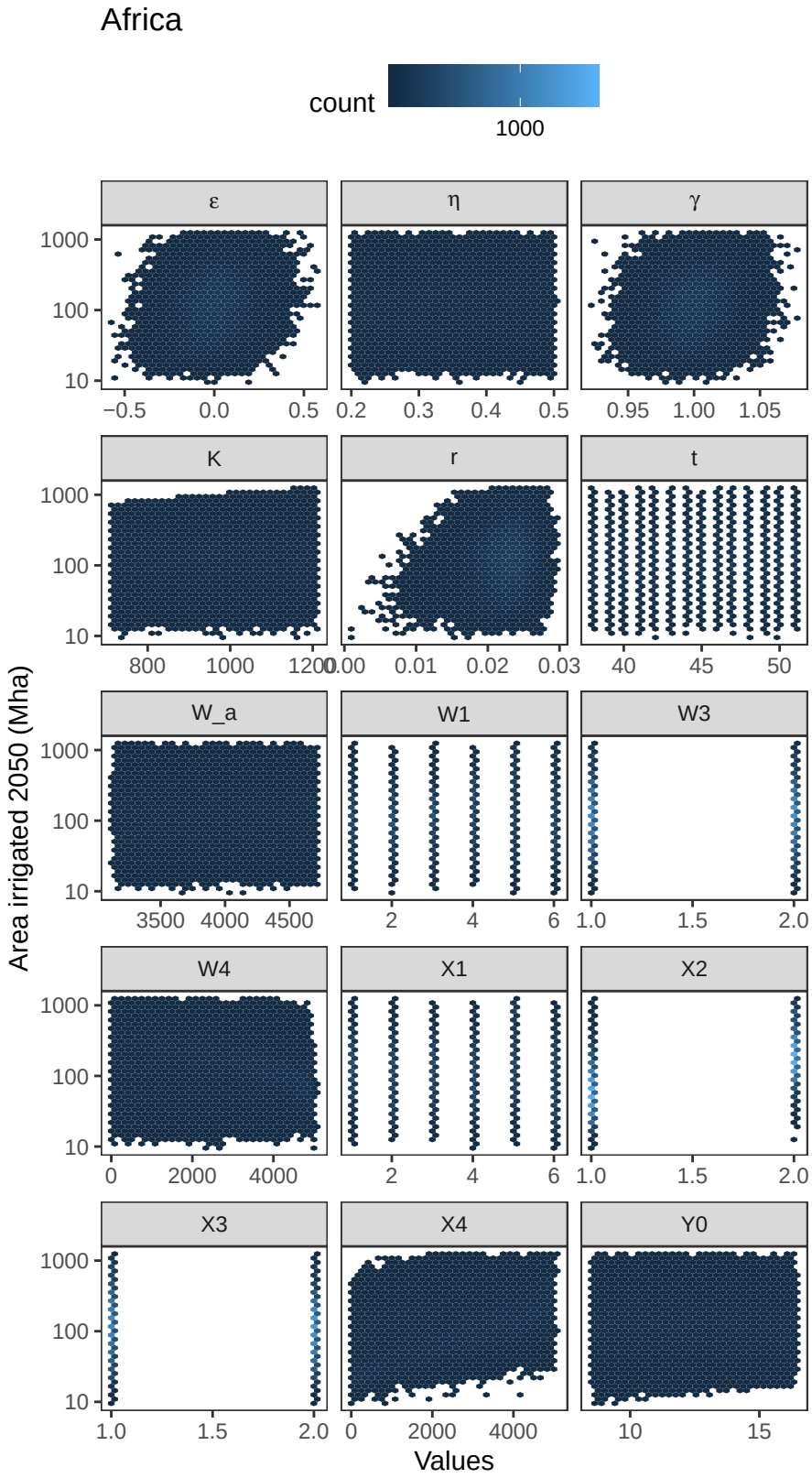
9 Sensitivity analysis

9.1 Scatterplots

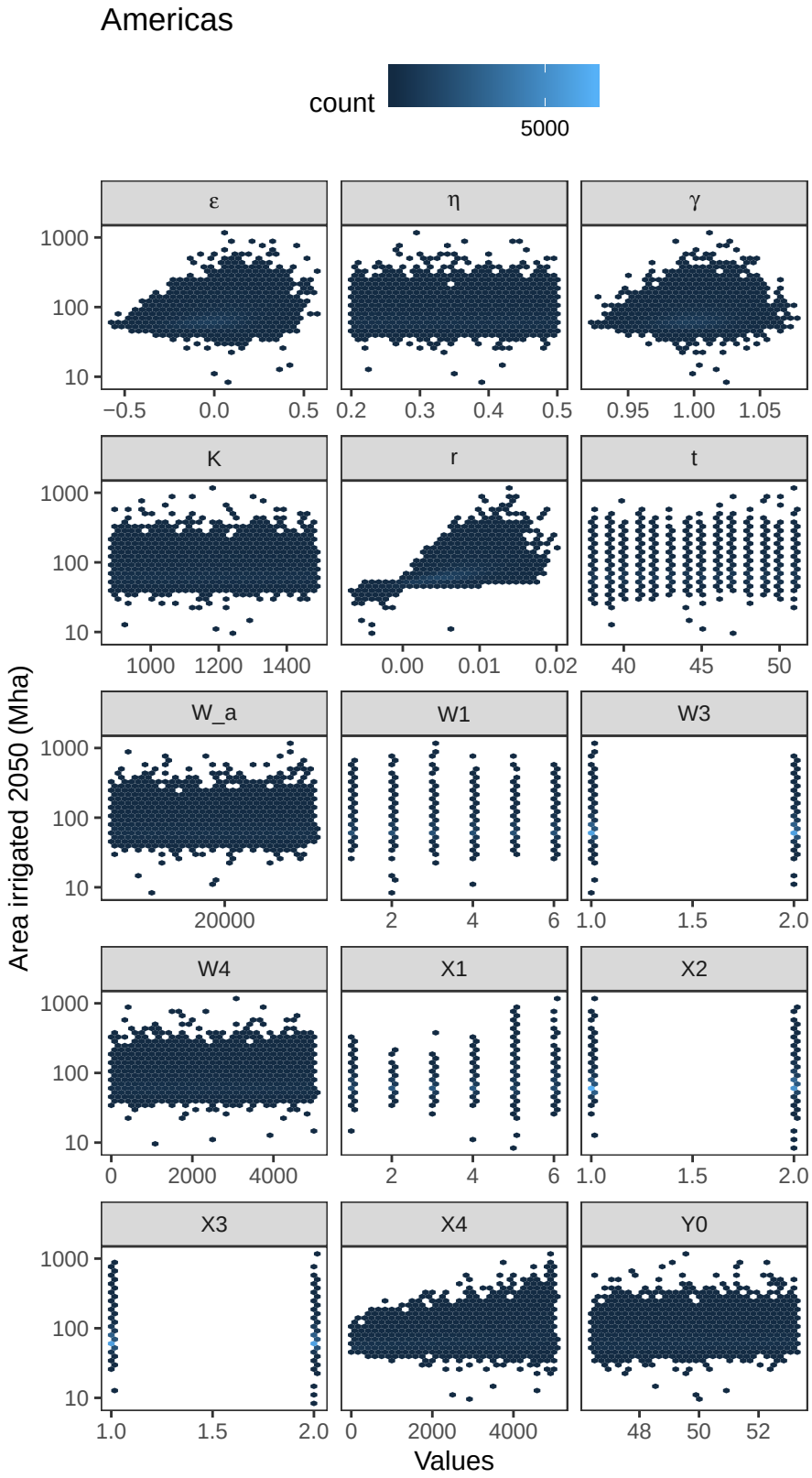
```
# ARRANGE SCATTERPLOTS OF PARAMETERS VS MODEL OUTPUT -------------------------

# Function to recode some parameters to allow plotting
code_columns <- function(x) {
  dt <- ifelse(x == "Aquastat", 1,
               ifelse(x == "FAOSTAT", 2,
                      ifelse(x == "Siebert.et.al.2013", 3,
                             ifelse(x == "Meier.et.al.2018", 4,
                                    ifelse(x == "Salmon.et.al.2015", 5, 6)))))
  return(dt)
}

code_columns2 <- function(x) ifelse(x == "YES", 1, 2)

# Vector with renamed parameters for better plotting
parameters.renamed <- c("X1", "X2", "X3", "X4", "W1", "W3", "W4", "r", "$\\gamma$",
                        "Y0", "$\\epsilon$", "t", "K", "Wa", "$\\eta$")

# Create temporary data table to plot
tmp <- cbind(AB.dt[, .(Continent)], AB.dt[, ..parameters], AB.dt[, .(Y)])

# Update columns and arrange
# Update columns to allow plotting of scatterplots
col_names <- c("X1", "W1")
col_names2 <- c("X3", "W3")
tmp <- tmp[, (col_names):= lapply(.SD, code_columns), .SDcols = col_names]
tmp <- tmp[, (col_names2):= lapply(.SD, code_columns2), .SDcols = col_names2]
tmp <- tmp[, X2:= ifelse(X2 == "OLS", 1, 2)]

tmp2 <- gather(tmp, Parameters, Values, X1:eta) %>%
  split(., .$Continent)
```

```
# PLOT SCATTERPLOTS OF PARAMETERS VS MODEL OUTPUT ---------------------------

gg <- list()
for(i in names(tmp2)) {
  gg[[i]] <- ggplot(tmp2[[i]], aes(Values, Y)) +
    geom_hex() +
    scale_x_continuous(breaks = pretty_breaks(n = 3)) +
    scale_fill_gradient(breaks = pretty_breaks(n = 2)) +
    scale_y_log10() +
    scale_alpha(guide = "none") +
    labs(x = "Values",
         y = "Area irrigated 2050 (Mha)") +
```

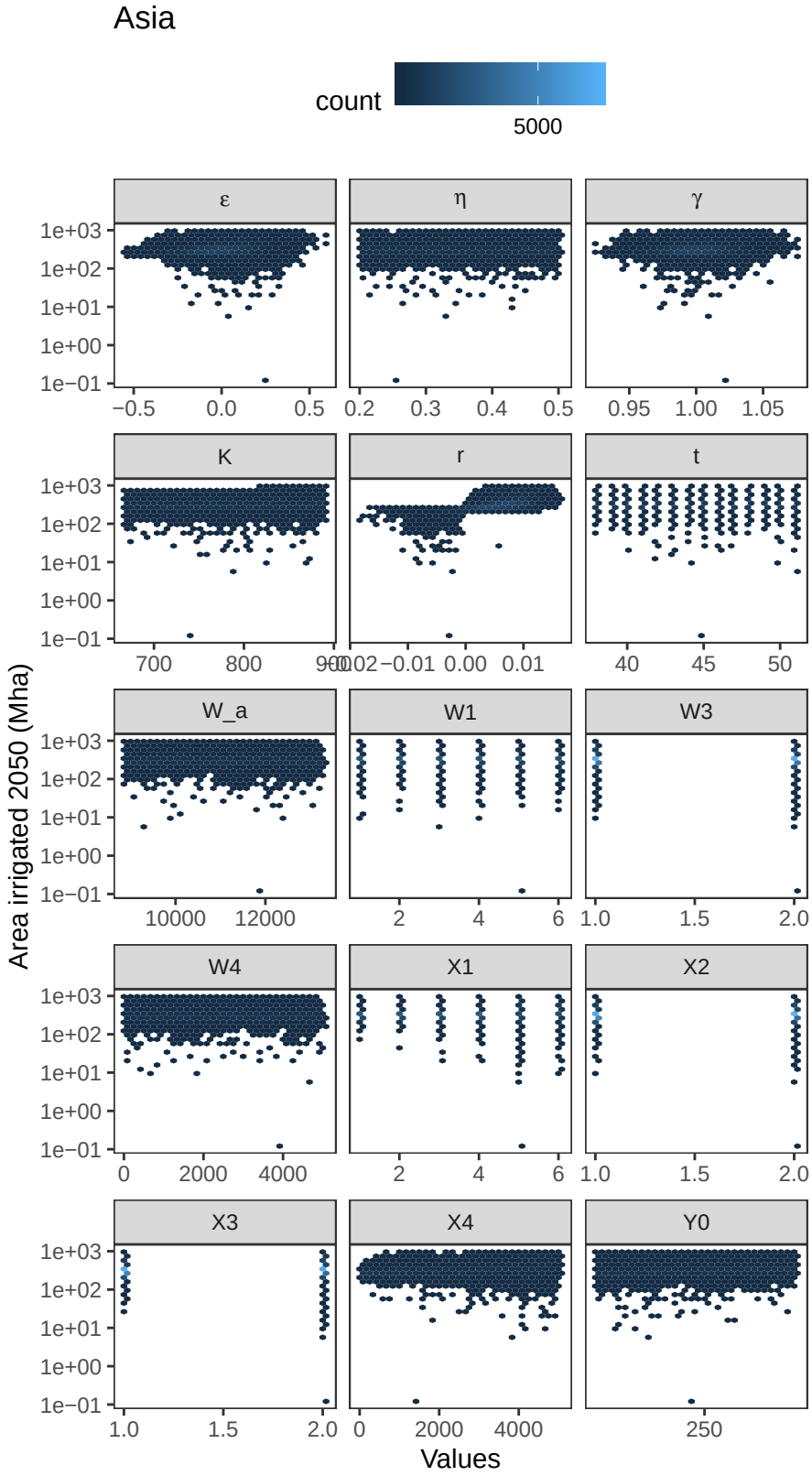
```
    facet_wrap(~Parameters,
               scales = "free_x",
               ncol = 3,
               labeller = label_parsed) +
    theme_bw() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.background = element_rect(fill = "transparent",
                                           color = NA),
          legend.key = element_rect(fill = "transparent",
                                    color = NA),
          legend.position = "top") +
    ggtitle(names(tmp2[i]))
}

gg[[1]]
```


Africa
count
1000
ε
η
γ
K
r
t
W_a
W1
W3
W4
X1
X2
X3
X4
Y0
Area irrigated 2050 (Mha)
Values

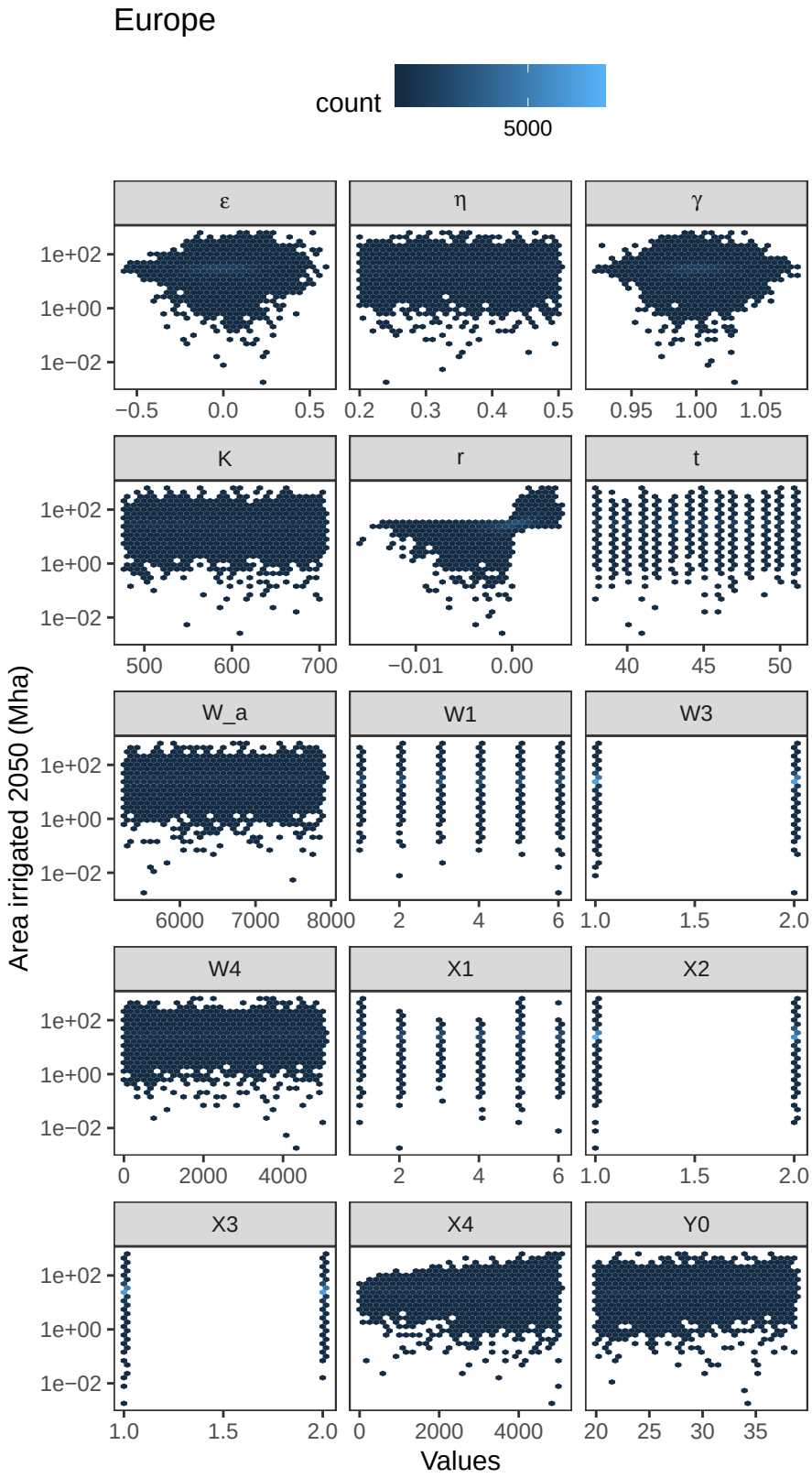
```
gg[[2]]
```


Americas
count
5000
ε
η
γ
K
r
t
W_a
W1
W3
W4
X1
X2
X3
X4
Y0
Area irrigated 2050 (Mha)
Values

```
gg[[3]]
```


Asia
count
5000
ε
η
γ
K
r
t
W_a
W1
W3
W4
X1
X2
X3
X4
Y0
Area irrigated 2050 (Mha)
Values

```
gg[[4]]
```


Europe
count
5000
ε
η
γ
K
r
t
W_a
W1
W3
W4
X1
X2
X3
X4
Y0
1e+02
1e+00
1e−02
Area irrigated 2050 (Mha)
Values

9.2 Sobol' indices

```
# SETTING FOR SOBOL' INDICES ----------------------------------------------

# Set the number of bootstraps
R <- 1000

# Set the confidence interval method
type <- "norm"

# Set the confidence interval
conf <- 0.95

# Create vector with the name of the clusters
cluster <- c("Irrigation", "Population", "Model")
```

```
# COMPUTE SOBOL' INDICES --------------------------------------------------

# Compute Sobol' indices
out <- full.dt[, sobol_indices(Y,
                               params = c(parameters.renamed, cluster),
                               type = "saltelli",
                               R = R,
                               n = n,
                               parallel = "multicore",
                               ncpus = floor(detectCores() * 0.75)),
               by = Continent]
```

```
## Warning in selectChildren(ac[!fin], -1): error 'No child processes' in select
```

```
# SOBOL' CONFIDENCE INTERVALS ---------------------------------------------

# Compute confidence intervals
tmp <- split(out, out$Continent)
out.ci <- list()
for(i in names(tmp)) {
  out.ci[[i]] <- sobol_ci(tmp[[i]],
                          params = c(parameters.renamed, cluster),
                          type = type,
                          conf = conf)
}
```

```
# SOBOL' INDICES OF A DUMMY PARAMETER -------------------------------------

# For the model parameters
out.dummy <- full.dt[, .SD[1:(n * (k + 2))], Continent] %>%
  .[, sobol_dummy(Y,
                  params = parameters.renamed,
                  R = R,
```

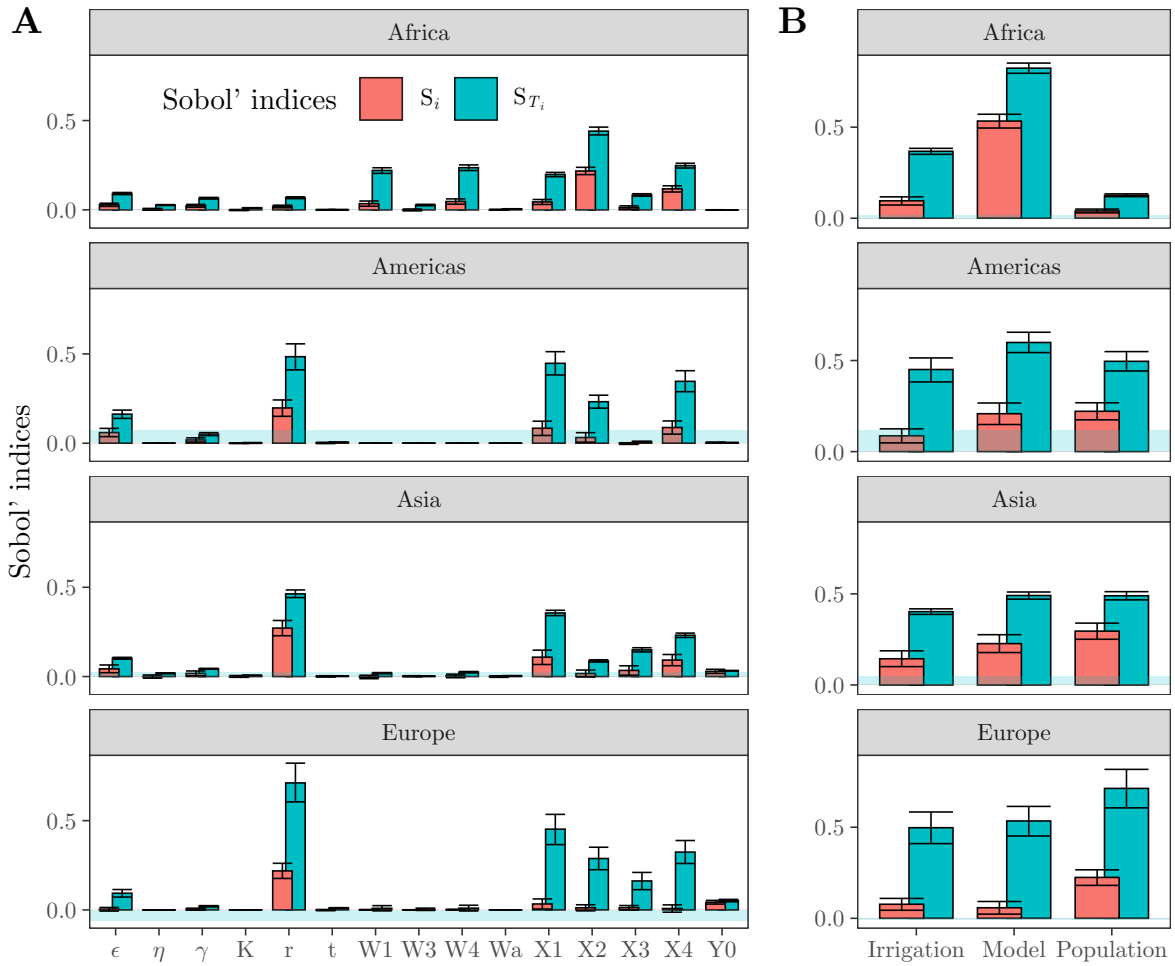
```
                  n = n,
                  parallel = "multicore",
                  ncpus = floor(detectCores() * 0.5)),
    by = Continent]

# Compute confidence intervals
tmp.dummy <- split(out.dummy, out.dummy$Continent)
out.dummy.ci <- list()
for(i in names(tmp.dummy)) {
  out.dummy.ci[[i]] <- sobol_ci_dummy(tmp.dummy[[i]],
                                      type = type,
                                      conf = conf)
}
out.dummy.ci2 <- rbindlist(out.dummy.ci, idcol = "Continent")

# For the clusters of parameters
tmp1 <- full.dt[, .SD[1:(2 * n)], Continent] %>%
  split(., .$Continent)

tmp2 <- full.dt[full.dt[, tail(.I, length(cluster) * n), by = Continent]$V1, ] %>%
  split(., .$Continent)

for(i in names(tmp1)) {
  tmp1[[i]] <- rbind(tmp1[[i]], tmp2[[i]])
}
out.dummy.cluster <- rbindlist(tmp1) %>%
  .[, sobol_dummy(Y,
                  params = cluster,
                  R = R,
                  n = n,
                  parallel = "multicore",
                  ncpus = floor(detectCores() * 0.5)),
    by = Continent]

# Compute confidence intervals
tmp.dummy <- split(out.dummy.cluster, out.dummy.cluster$Continent)
out.dummy.cluster.ci <- list()
for(i in names(tmp.dummy)) {
  out.dummy.cluster.ci[[i]] <- sobol_ci_dummy(tmp.dummy[[i]],
                                              type = type,
                                              conf = conf)
}

out.dummy.cluster.ci2 <- rbindlist(out.dummy.cluster.ci, idcol = "Continent")
```

```
# EXPORT SOBOL' INDICES ------------------------------------------------------
```

```r
sobol.ci <- rbindlist(out.ci, idcol = "Continent")
fwrite(sobol.ci, "sobol.ci.csv")
```

```r
# PREPARE PLOT SOBOL' INDICES ------------------------------------------------

# Plot Sobol' indices of parameters
a <- rbindlist(out.ci, idcol = "Continent") %>%
  .[!parameters %in% cluster] %>%
  plot_sobol(., type = 1, dummy = out.dummy.ci2) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  facet_wrap(~Continent, ncol = 1) +
  labs(y = "Sobol' indices",
       x = "" )

# Plot Sobol' indices of clusters of parameters
b <- rbindlist(out.ci, idcol = "Continent") %>%
  .[parameters %in% cluster] %>%
  plot_sobol(., type = 1, dummy = out.dummy.cluster.ci2) +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  facet_wrap(~Continent, ncol = 1) +
  labs(x = "",
       y = "")
```

```r
# PLOT SOBOL' INDICES --------------------------------------------------------

# Merge legend and a and b
plot_grid(a + theme(legend.position = c(0.4, 0.95),
                    legend.direction = "horizontal"),
                    b + theme(legend.position="none"),
                    ncol = 2,
                    labels = "AUTO",
                    rel_widths = c(2, 1.1))
```




```
# Merge both plots
bottom <- plot_grid(a + theme(legend.position="none",
                              axis.text.x = element_text(size = 7.2)),
                    b + theme(legend.position="none",
                              axis.text.x = element_text(size = 7.2)),
                    ncol = 2,
                    labels = "AUTO",
                    rel_widths = c(2, 1.1))

plot_grid(legend,
          bottom,
          ncol = 1,
          rel_heights = c(0.15, 1))
```


```
# CHECK SUM OF SI INDICES ----------------------------------------------------

sobol.ci[parameters %in% c("Irrigation", "Population", "Model")] %>%
  .[sensitivity == "Si"] %>%
  .[, sum(original),  Continent]
```

```
##    Continent        V1
## 1:    Africa 0.6690779
## 2:  Americas 0.5160329
## 3:      Asia 0.6665734
## 4:    Europe 0.3597039
```

10 Session information

```
# SESSION INFORMATION ----------------------------------------------------------

sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.3
##
```

```
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] checkpoint_0.4.8    sensobol_0.2.1      NbClust_3.0
##  [4] gridExtra_2.3       wesanderson_0.3.6   sensitivity_1.17.0
##  [7] doParallel_1.0.15   iterators_1.0.12    foreach_1.4.7
## [10] boot_1.3-24         smatr_3.4-8         robustbase_0.93-5
## [13] randtoolbox_1.30.0  rngWELL_0.10-6      complmrob_0.7.0
## [16] mvoutlier_2.0.9     sgeostat_1.0-27     cowplot_1.0.0
## [19] forcats_0.4.0       stringr_1.4.0       dplyr_0.8.3
## [22] purrr_0.3.3         readr_1.3.1         tidyr_1.0.0
## [25] tibble_2.1.3        ggplot2_3.2.1       tidyverse_1.3.0
## [28] scales_1.1.0        countrycode_1.1.0   readxl_1.3.1
## [31] fGarch_3042.83.1    fBasics_3042.89     timeSeries_3042.102
## [34] timeDate_3043.102   fitdistrplus_1.0-14 npsurv_0.4-0
## [37] lsei_1.2-0          survival_3.1-8      MASS_7.3-51.5
## [40] data.table_1.12.8
##
## loaded via a namespace (and not attached):
##  [1] backports_1.1.5       plyr_1.8.5            lazyeval_0.2.2
##  [4] sp_1.3-2              splines_3.6.1         digest_0.6.23
##  [7] htmltools_0.4.0       fansi_0.4.1           magrittr_1.5
## [10] cluster_2.1.0         openxlsx_4.1.4        modelr_0.1.5
## [13] colorspace_1.4-1      rvest_0.3.5           rrcov_1.5-2
## [16] haven_2.2.0           xfun_0.12             crayon_1.3.4
## [19] jsonlite_1.6          zeallot_0.1.0         zoo_1.8-7
## [22] glue_1.3.1            gtable_0.3.0          car_3.0-6
## [25] kernlab_0.9-29        prabclus_2.3-2        DEoptimR_1.0-8
## [28] abind_1.4-5           VIM_4.8.0             mvtnorm_1.0-12
## [31] DBI_1.1.0             GGally_1.4.0          bibtex_0.4.2.2
## [34] Rcpp_1.0.3            sROC_0.1-2            laeken_0.5.0
## [37] foreign_0.8-75        mclust_5.4.5          stats4_3.6.1
## [40] vcd_1.4-5             truncnorm_1.0-8       httr_1.4.1
## [43] RColorBrewer_1.1-2    fpc_2.2-4             modeltools_0.2-22
## [46] spatial_7.3-11        pkgconfig_2.0.3       reshape_0.8.8
## [49] NADA_1.6-1            flexmix_2.3-15        nnet_7.3-12
## [52] dbplyr_1.4.2          tidyselect_0.2.5      rlang_0.4.2
## [55] munsell_0.5.0         cellranger_1.1.0      tools_3.6.1
## [58] cli_2.0.1             generics_0.0.2        ranger_0.12.1
```

```
## [61] pls_2.7-2            broom_0.5.3           evaluate_0.14
## [64] cvTools_0.3.2        yaml_2.2.0            knitr_1.27
## [67] fs_1.3.1             zip_2.0.4             nlme_3.1-143
## [70] xml2_1.2.2           compiler_3.6.1        rstudioapi_0.10
## [73] curl_4.3             e1071_1.7-3           zCompositions_1.3.3-1
## [76] reprex_0.3.0         robCompositions_2.2.0 pcaPP_1.9-73
## [79] stringi_1.4.5        lattice_0.20-38       Matrix_1.2-18
## [82] vctrs_0.2.1          pillar_1.4.3          lifecycle_0.1.0
## [85] Rdpack_0.11-1        lmtest_0.9-37         gbRd_0.4-11
## [88] R6_2.4.1             rio_0.5.16            codetools_0.2-16
## [91] assertthat_0.2.1     withr_2.1.2           diptest_0.75-7
## [94] hms_0.5.3            class_7.3-15          rmarkdown_2.1
## [97] carData_3.0-3        lubridate_1.7.4
```


FAO. 2016. "AQUASTAT website." Food; Agriculture Organization of the United Nations. http://www.fao.org/nr/water/aquastat/didyouknow/index3.stm.

———. 2017. "FAOSTAT database." Rome. http://www.fao.org/faostat/en/.

Meier, J., F. Zabel, and W. Mauser. 2018. "A global approach to estimate irrigated areas . A comparison between different data and statistics." *Hydrology and Earth System Sciences* 22 (2): 1119–33. https://doi.org/10.5194/hess-22-1119-2018.

Salmon, J.M., M. A. Friedl, S. Frolking, D. Wisser, and E. M. Douglas. 2015. "Global rain-fed, irrigated, and paddy croplands: A new high resolution map derived from remote sensing, crop inventories and climate data." *International Journal of Applied Earth Observation and Geoinformation* 38. Elsevier B.V.: 321–34. https://doi.org/10.1016/j.jag.2015.01.014.

Siebert, S., V. Henrich, K. Frenken, and J. Burke. 2013. "Update of the digital global map of irrigation areas to version 5." Rome: Rheinische Friedrich-Wilhelms-University; Food; Agriculture Organization of the United Nations.

Thenkabail, P. S., C. M. Biradar, P. Noojipady, V. Dheeravath, Y. Li, M. Velpuri, M. Gumma, et al. 2009. "Global irrigated area map (GIAM), derived from remote sensing, for the end of the last millennium." *International Journal of Remote Sensing* 30 (14): 3679–3733. https://doi.org/10.1080/01431160802698919.